UNIX 사용자편람

김일성종합대학출판사
주체91

UNIX
사 용 자 편 람

김일성종합대학출판사

차 례

제3장. 파일과 등록부에 대한 작업 - 허락, 지령, 파일이름확장, 통용기호

제４장. 파일보기 - 방향바꾸기, cat, more, pg, head, tail지령

제５장. UNIX도구 - split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, tr지령

제6장. 개선된 UNIX도구 - 정규식, sed, awk, grep지령

제7장. find 지령

제8장. vi편집기

제9장. Bash쉘에 대한 개괄

제１０장. Korn쉘에 대한 개괄

제１１장. C쉘에 대한 개괄

제１２장. 쉘프로그람작성에 대한 개괄

제１３장. 체계관리에 대한 개괄

제１４장. UNIX수행도구 들에 대한 개괄

제１５장. 공동탁상환경

제１６장. 망

프로그람작성편

제１７장. 쏘프트웨어 개발의 기초

제１８장. 프로그람작성의 구성체

제１９장. 프로그람작성 설계

제２０장. 개 발

제２１장. C와 C++에 대한 개괄

제２２장. C프로그람작성의 기초

제２３장. C++프로그람 작성의 기초

제２６장. Perl에 대한 소개

Unix와 Windows의 호상작용편

제２７장. X Windows체계

제31장. UNIX를 위한 봉사(SFU)

제３２장. Samba

색인

머 리 글

UNIX 는 눈에 띄지 않게 연구되고 개발되였지만 오늘은 세계적인 인터네트에서 광범히 리용되고 있다. 과거로부터 UNIX 가 얼마나 많이 변화되여 왔는가를 상상해 보기란 아름찬 일이다. 그러나 콤퓨터분야에서 이러한 갱신은 시간이 그리 오래 걸리는 일이 아니다.

UNIX 조작체계는 연구와 개발로부터 시작하여 일정한 주요산업부문에서의 리용을 거쳐 국제적인 인터네트에까지 진보하여 왔다. 오늘날 콤퓨터분야에서의 UNIX 체계는 비 UNIX 체계에서의 UNIX 명령모임의 리용으로부터 여러가지 Linux 배포판들과 HP-UX, Solaris, AIX 와 같은 각이한 UNIX 변종들에 이르기까지 넓은 범위의 기능들을 포괄하고 있다. UNIX 는 전통적인 콤퓨터실들에 설치되고 있을뿐아니라 탁상형콤퓨터, 노트형콤퓨터, PDA 등에서도 찾아 볼수 있다.

UNIX 조작체계의 매개 과제와 절차들을 리해하는것이 결코 중요한것은 아니다. 오늘과 같은 콤퓨터시대에서는 체계관리자이건 초학자 혹은 경험 있는 사용자이건 누구나다 UNIX 와 Linux 에 대한 개념과 그 실현에 대한 기본적인 지식을 가져야 한다.

많은 도서들과 쏘프트웨어 그리고 참고서들에서는 정의를 정확히 내리지 않은채로 UNIX 에 대한 기본지식을 론의하고 있다. 《손바닥에 있는 한마리의 새가 숲속에 있는 두마리의 새보다 낫다.》는 식으로 학습을 하는데서 언제나 한계를 명백히 긋고 고찰해 나가면 더 쉽기때문에 이 책에서는 기본적인 UNIX 지식에 대하여 다음과 같이 정의하였다. 즉 UNIX 에 대한 기본적인 리해를 가지려면 UNIX 의 구조, 가입 및 탈퇴방법, 등록부이동에 필요한 지령들, 편집기, 일부 쉘지령들, 인터네트기초, 전자우편프로그람, 쉘프로그람작성의 원리, C 언어에 대하여 어느 정도 알아야 한다. 그러나 받은 지식은 한쪼각의 도마도를 먹는것과 같으므로 대부분의 독자들과 사용자들은 더 많은 UNIX 에 대한 참고서들을 읽고 학습하는 과정을 통하여 UNIX 에 대한 완벽한 지식과 경험을 쌓아야 한다.

일부 문서들은 유감스럽게도 비데오록음기(VCR)에 대한 사용지도서와 비슷하다. 이런 문헌들은 참고가 되기는 하지만 만족할만한것은 못된다. 훌륭한 콤퓨터문헌들은 배우려는 개념을 직관적으로 잘 설명하고 있을뿐아니라 6 개월만에 필요한 모든 정보를 다 얻을수 있게 하고 있다.

이 책에서는 UNIX 조작체계의 구조를 인식시키는 방법으로 설명하며 그것의 지령들과 3 개의 기본적인 쉘환경, vi 편집기, 자료조종수단들, 공동탁상환경(CDE), 프로그람개발방법들에 대하여서도 자세히 주고 있다. 이 책의 내용을 통하여 독자들은 체계의 관리와 개발, Linux 까지 포함하여 여러가지 UNIX 변종들에서 실례를 통한 실천능력을 키울수 있다. 많은 장들은 그 장에서 쓰이는 지령들에 대한 지령소개들로 끝을 맺고 있다. 추가적인 고속접근자원으로서 vi 참조카드도 제공되여 있다. 새로운것은 Perl 에 대한 기초를 개괄하고 있는 장이다. 결국 이 책은 초학자 및 경험 있는 사용자 그리고 체계관리자들을 위한 참고서로 된다.

제1장. 첫 시작 - 가입, 우편, 인터네트접근, UNIX 구성요소

이 장에서는 UNIX 체계의 지령들을 사용한 실례들을 통하여 UNIX 체계를 빨리 리해하기 위한 여러가지 항목들에 대하여 설명한다. 이 책에서 독자들은 UNIX 변종들사이의 차이점들을 보게 되겠지만 그에 대하여 지나치게 관심할 필요는 없다. 례를 들어 어떤 UNIX 체계의 지령이 다른 UNIX 변종들의것과 다를수 있지만 그것의 존재성만 알면 충분하다.

CDE 는 많은 UNIX 변종들에서 리용하는 도형사용자대면부로서 **공동탁상환경**(Common Desktop Environment)이라고 부른다. CDE 는 자기의 체계에서 나타나지 않을수도 있으며 체계관리자가 다른 도형사용자대면부를 선택할수도 있다. 이 책에서 고찰하는 항목들은 대체로 모든 변종들에서 동일하므로 이 장에서 주는 대부분의 실례를 개별적변종들에서도 리용할수 있을것이다. 일반적으로 지령재촉상태에서 진행되는 일들은 다른 UNIX 변종들에서도 류사하다.

이 장의 제일 마지막부분에서는 UNIX 성분들에 대하여 고찰한다. 이 부분을 뒤에서 고찰하는 리유는 사용자들이 우선 UNIX 체계에 가입한 다음 기본적인 지령들을 실행시켜 본 후에야 그 진가를 더 잘 알수 있기때문이다.

다중사용자체계 UNIX

대부분의 UNIX 체계들은 **다중사용자체계**(Multi-User-System)이다. 그 의미는 하나의 콤퓨터를 여러 사용자가 동시에 리용할수 있다는것이다. UNIX 체계에는 여러가지 방법으로 접속할수 있다. 도형대면부를 지원하지 않는 체계에서는 지령행에서 지령을 입력하는 기호말단장치로 접속할수도 있고 콤퓨터에 직접 련결된 도형현시장치가 달린 콤퓨터로 접속할수도 있다. 또한 국부망(LAN)에서 도형사용자대면부를 동작시키는 X-말단장치로도 접속할수 있다. 여기서 X-말단장치와 콤퓨터에 직접 련결된 도형현시장치는 둘다 도형사용자대면부 즉 공동탁상환경(보통 CDE)을 동작시키기때문에 직접 식별하기는 어렵다. UNIX 변종들에 여러가지 대면부가 있다고 하여도 사용자는 항상 지령행으로 UNIX 에 접근할수 있다. 결국 넓은 의미에서 보면 도형대면부와 도형장치가 존재하는데 그것들은 모두 지령행으로 사용자를 접근시켜 주는 말단창문들을 제공할뿐이다.

기호에 기초한 혹은 도형에 기초한 한개 말단만이 접속되여 있는 체계들도 보통 다중사용자로 될수 있다. 그렇다면 련결된 하나의 말단을 가지고 무엇때문에 다중사용자들을 지원할 필요가 있겠는가? 그것은 직접 련결된 말단장치에서 작업하는 한명의 사용자와 함께 국부망을 통하여 련결된 여러명의 사용자들이 있을수 있기때문이다. 또한 서로 다른 사용자들의 각이한 과제를 수행하는 다중창문들이 도형말단에서 열리게 할수도 있기때문이다.

단일사용자체계(Single-User System)들은 보통 개별적인 사용자들을 위한것이다. 이러

한 체계들은 UNIX 환경전부를 개별적인 사용자에게 제공하지만 그것은 어디까지나 한명의 사용자에게만 접근된다. 이 책에서 고찰하는 문제들은 단일사용자체계 및 다중사용자체계에 다 적용되는것들이다.

모든 UNIX 체계들은 하나의 체계에서 동시에 여러개의 과제 및 응용프로그람이 수행된다는 의미에서의 **다중과제체계**(Multi-Task System)이다. UNIX 체계에서 동시에 많은 과제들이 수행되는데 대하여서는 이 책의 뒤부분에서 고찰한다.

대문자 및 소문자

UNIX 체계는 지령과 파일이름에서 대문자와 소문자를 구별한다. 즉 대소문자에 의하여 지령이나 파일이름의 의미는 서로 달라 진다. 다음의 실례는 UNIX 체계에서 서로 다른것으로 식별되는 파일이름들이다.

```
program
Program
prograM
PROGRAM
```

program 이라는 파일을 UNIX 체계에서 cc 지령으로 콤파일하자면 다음과 같이 지령을 주어야 한다.

```
cc  program
```

만일 다음의 지령들중의 어느 하나처럼 지령을 주면 목적하는 프로그람을 콤파일할수 없다.

```
CC  program
cc  Program
```

첫번째 실례에서 CC 는 대문자이므로 소문자들로 이루어 진 cc 와는 다른 지령이다. 그러므로 목적하는 프로그람을 실행시키지 못한다. 두번째 실례에서는 Program 이 목적하는 프로그람인 program 이 아니므로 제대로 수행할수 없다.

UNIX 체계에서 대소문자를 구별하는데 주의하면서 보충적으로 알아야 할것은 UNIX 체계에 여러가지 파일종류가 있다는것이다. 파일종류가 어떻게 구분되는가에 대하여서는 제 2 장에서 고찰한다.

가입절차

UNIX 체계의 모든 사용자들은 체계관리자에 의하여 설정된 **가입이름**(Login Name)을 가진다. 체계관리자는 사용자가 희망하는 가입이름으로 사용자를 받아 들인다. 사용자는 가입이름과 통과암호를 가지고 체계에 대한 접근허가를 얻는다. 이 절차에 대하여 간단히 보자.

사용자가 가입이름을 선택하는 규칙은 체계관리자에 의하여 결정된다. 일반적으로 가입이름은 2−8 개까지의 기호렬이 될수 있다. 가입이름에는 특수기호들이 리용될수 있다.

가입이름에 대한 보안대책이 없기때문에 자기가 기억하기 쉽게 이름을 따내면 된다. 가입이름은 체계를 리용하는 다른 사용자들로부터 자기에게 통보나 전자우편을 보내올 때 리용되며 그 가입이름에 의하여 다른 사용자들이 자기를 알게 된다.

통과암호(Password)는 체계관리자가 설정한 요구조건에 반드시 맞아야 한다. 처음으로 가입할 때에는 통과암호가 없거나 림시통과암호만을 가질수 있다. **림시통과암호**(Temporary Password)는 체계에 처음으로 가입한 이후 변경될수 있다. 통과암호는 표준적으로 최소 6 개의 기호와 최소 한개의 특수기호를 가진다. 체계관리자는 통과암호가 규칙적으로 변경되도록 할수 있다. 통과암호는 의미가 없게 작성되여야 한다. 그래서 누구도 그것을 알아 맞추기 위하여 시도조차 하지 못하게 하여야 한다.

이제 가입절차에 대한 실례를 보기로 하자. 기호에 기초한 가입과정으로 시작한다. 말단장치에서 다음의 실례와 같은 **가입재촉문**(Login Prompt)을 받게 된다. 다음 가입신청자는 체계관리자로부터 자기가 받은 이름과 림시통과암호로 대답한다.

```
*****************************************************
* This is a private system operated for Your Company       *
* business. Authorization from management is required to use *
* this system. Use by unauthorized persons is prohibited.     *
*****************************************************
```

(이것은 당신의 회사를 위해서 움직이는 개별체계입니다. 이 체계를 리용하려면 관리자로부터 권한을 부여 받아야 합니다. 권한을 부여 받지 못한 사람이 리용하는것은 금지되여 있습니다.)

login: **martyp**
Password:
Last login: Fri Sep 17 07:12:57 from atlm0547.atl.hp.
TERM set to vt100

martyp $

가입이 실현되였다. 정확한 가입이름과 통과암호를 입력하였으며 그리고 체계에 대

한 접근을 허락 받았다.

혹시 사용자이름과 통과암호를 틀리게 입력할수도 있다. 다음의 실례에서는 우선 유효한 사용자이름과 무효한 통과암호를 입력하였다. 다음으로 무효한 사용자이름을 입력하였다. 세번째에는 유효한 정보를 입력하였으므로 체계에 대한 접근을 허락 받는다.

```
***************************************************
*  This is a private system operated for Your Company        *
*  business. Authorization from management is required to use    *
*  this system. Use by unauthorized persons is prohibited.       *
***************************************************

login: martyp
Password:                      ← 첫번째 시도, 무효한 통과암호
Login incorrect

***************************************************
*  This is a private system operated for Your Company        *
*  business. Authorization from management is required to use    *
*  this system. Use by unauthorized persons is prohibited.       *
***************************************************

login: m                       ← 두번째 시도, 무효한 사용자이름
Password:
Login incorrect

***************************************************
*  This is a private system operated for Your Company        *
*  business. Authorization from management is required to use    *
*  this system. Use by unauthorized persons is prohibited.       *
***************************************************

login: martyp                  ← 세번째 시도, 유효한 사용자이름과 통과암호
Password:
Last login: Fri Sep 17 07:12:57 from atlm0547.atl.hp.
DISPLAY set to atlm0547:0.0

TERM set to vt100

martyp $
```

우의 실례에서 첫번째와 두번째 가입시도는 실패하였다. 체계는 첫번째 시도에서는 틀린 통과암호를 인식하였고 두번째 시도에서는 틀린 사용자이름을 인식하였으므로 두 경우에 모두 "Login incorrect"라는 통보를 내보내고 시동을 하지 않았다. 결국 "Login incorrect"라는 통보는 사용자이름과 통과암호중의 어느것이 잘못되였는가를 명백히 구분하여 주지 않으므로 모호하다. 어쨌든 어느 하나 혹은 둘 다 틀리게 입력되였다는것을 의미한다.

가입이 성공하면 체계와 자기의 말단 그리고 체계관리자에 대한 통보를 받게 되는데 이 통보는 체계에 따라 약간 다를수 있다. 가입후에는 지령들을 줄수 있다. 실례로 첫 지령은 체계관리자가 제공한 림시통과암호를 변경시키게 할수 있다.

passwd 지령은 통과암호를 변경시키는 지령이다. 먼저 passwd 지령으로 자기의 암호를 현재 암호로 변경시켜 보자. 즉 아무런 변화가 없게 한다. 그다음 통과암호를 passwd 라고 변경시키자. 마지막으로 유효한 암호로 변경시켜 보자.

martyp $ **passwd**
passwd: Changing password for martyp
Enter login password:
New password: **← 현재의 통과암호로 변경한다**
passwd（SYSTEM）: Password cannot be circular shift of logonid.
New password: **← passwd 로 변경한다**
passwd(SYSTEM): The first 6 characters of the password
must contain at least two alphabetic characters and at least
one numeric or special character.
New password: **← 유효한 통과암호로 변경한다**
Re-enter new password:
passwd（SYSTEM）: passwd successfully changed for martyp
martyp $

암호를 현재 통과암호로 변경시키려고 할 때 (현재 통암호로 한다는것은 사용자의 이름과 같게 한다는것이다.) "Passwd cannot be circular shift of logonid."라는 통보가 나온다. 이것은 가입이름과 통과암호가 이 체계에서는 달라야 한다는것을 의미한다. 다음으로 암호를 passwd 로 변경시키려고 할 때 첫 6 개의 기호는 적어도 2 개의 자모기호와 적어도 하나의 수자 혹은 특수기호를 포함하여야 한다는 통보가 나온다. 이것은 통과암호를 알아 맞추기 힘들게 하기 위해서이다. 마지막으로 통과암호를 정확히 입력하였을 때 변경이 진행되였다는 통보를 받게 된다. 체계는 새로운 통과암호를 입력할 때 볼수 없는 기호들로 표시하여 준다.

암호를 잘 만들기 위하여서는 다음의 특징을 살려야 한다.

- 최소 6 개기호들은 /, ., * 와 같은 특수기호를 포함한다.

- 단어들은 통과암호로 리용되지 말아야 한다.
- 암호를 자기이름이나 집주소 또는 널리 알려 진 체육단이름과 같은것으로 주지 말아야 한다.
- 123456 이나 qwerty 와 같이 입력하기 쉽게 주지 말아야 한다. 어떤 사람들은 맞춤법오유가 있는 단어는 받아 들일수 있다고 보는데 맞춤법검사프로그람을 동작시키면 알아 맞출수 있기때문에 좋은 방법이 아니다.
- 알아맞추기 어려운 암호를 생성해 내는 암호발생프로그람이 좋다.

passwd지령은 초기에 가입할 때 자기의 통과암호를 즉시 바꿀 필요가 있기때문에 알아 두어야 할 중요한 지령이다.

만일 도형사용자대면부의 가입화면을 통하여 체계에 접근한다면 지령행에서처럼 사용자이름과 통과암호를 입력하면 된다. 오유가 있으면 다시 입력하고 성공적인 가입후에는 도형작업환경을 가지게 된다. 그림 1-1은 많은 창문들이 열려 진 CDE를 보여 준다.

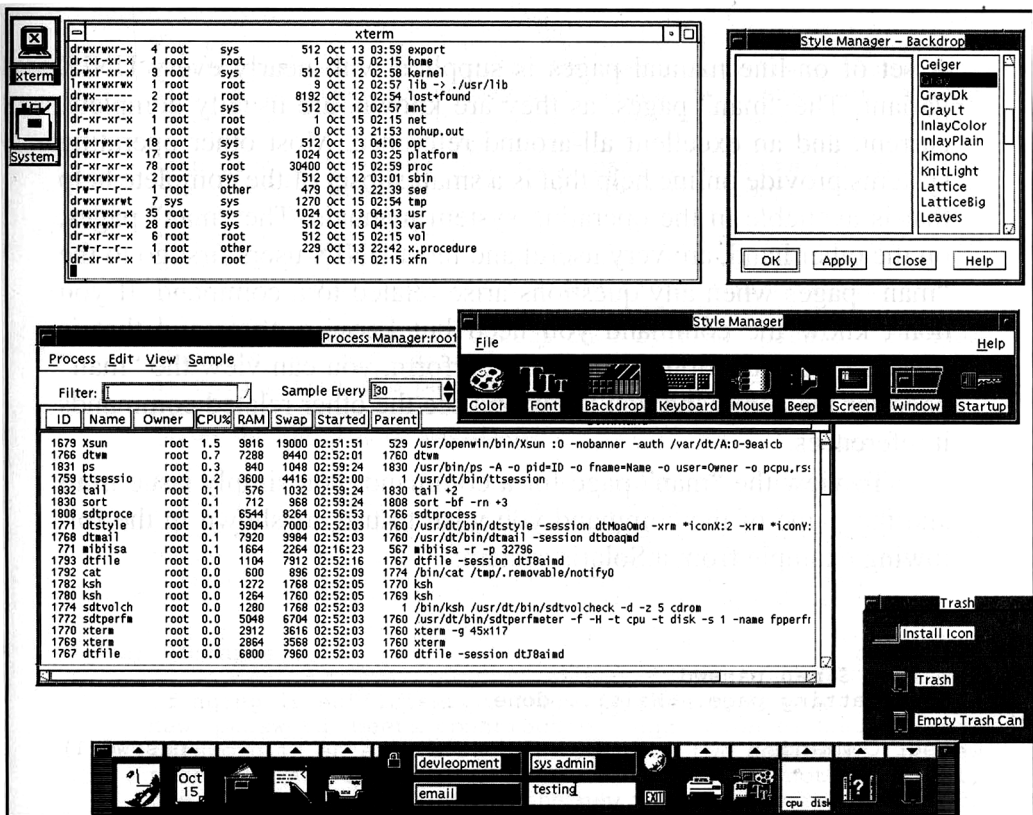



그림 1-1. 성과적인 가입후의 공동탁상환경

그림 1-1에서 보게 되는 매 구성요소들은 제 4 장에서 자세히 고찰한다.

직결안내페지

최근의 UNIX 변종에는 직결안내페지들이 제공되여 있다. 이 안내페지들은 우수하게 완성된 참고서이다. 대부분의 다른 조작체계들은 완성된 도움말(Help)이 조작체계안내서들에 있고 직결도움말에서는 그 일부 도움말만을 제공한다. 지령에 대하여 모를것이 있으면 처음으로 직결안내페지를 참고하게 된다. 그리고 해당한 지령은 모르지만 그 지령과 관련된 다른 지령을 알고 있다고 하면 그 관련지령의 지령소개에 가서 관련항목들을 보면 될것이다.

어떤 지령에 대한 지령소개("**man**"페지)를 보려면 간단히 man 이라고 입력하고 그뒤에 알고 싶은 지령이름을 주면 된다. 다음의 실례는 Solaris 체계에서 실행시킬수 있는것이다.

martyp $ **man passwd**
Reformatting page. Wait ... done

사용자지령 passwd(1)

이름

passwd – 가입통과암호와 통과암호속성을 변경시킨다.

표준일치

passwd [-r | files | -r nis | -r nisplus] [name]
passwd [-r files] [-egh] [name]
passwd [-r files] -s [-a]
passwd [-r files] -s [name]
passwd [-r files] [-d | -1] [-f] [-n min] [-w warn] [-x max] name
passwd -r nis [-egh] [name]
passwd -r nisplus [-egh] [-D domainname] [name]
passwd -r nisplus -s [-a]
passwd -r nisplus [-D domainname] -s [name]
passwd -r nisplus [-1] [-f] [-n min] [-w warn] [-x max] [-D domainname]
name

해설

지령 passwd 는 통과암호를 변경시키거나 사용자의 가입이름과 관련되는 통과암호속성들을 현시한다. 특정한 사용자들은 임의의 가입이름과 관련되는 통과암호와 속성들을 지령 passwd 를 사용하여 설치하거나 변경시킬수 있다.

통과암호를 변경시키기 위하여 이 지령을 사용할 때 passwd 는 낡은 통과암호에 대한

입력재촉문을 내보낸다. 그다음 새로운 통과암호를 두번 입력할것을 요구한다. 낡은 통과암호가 입력되였을 때 passwd 는 그것이 얼마나 리용되였는가를 검사하고 충분히 리용되지 못하였을 때에는 정지한다(pwcn(lM), nistbladm(1), shadow(4)참고).

NIS 혹은 NIS+가 가능한 체계라면 passwd 는 NIS 혹은 NIS+ 자료기지를 변경한다. NIS 혹은 NIS+ 통과암호는 국부기계의 통과암호와 다를수 있다. NIS 혹은 NIS+를 실행하고 있는 경우 passwd -r 를 리용하여 국부기계에 대한 통과암호정보를 변경시켜야 한다.

지령 pwconv 는 /etc/passwd 로부터 /etc/shadow 의 정보를 갱신한다. pwconv 는 /etc/passwd 의 통과암호마당에 있는 'x'의 특정한 값을 리용한다. 'x'의 값은 사용자의 통과암호가 /etc/shadow 에 이미 있으며 수정되지 말아야 한다는것을 지적한다.

사용자지령

통과암호가 충분히 사용되였을 경우 새로운 통과암호가 정확히 작성되였는가에 대한 검사가 진행된다. 새로운 통과암호가 두번째로 입력되였을 때 그 두개의 사본이 비교된다. 이 비교가 실패하면 새로운 통과암호에 대한 재촉이 두번 반복된다.

통과암호들은 다음과 같은 조건에 맞게 작성되여야 한다.

--More--(17%)

뒤에 붙어 있는 "17%"는 이 장의 앞에서 리용한 passwd 지령에 대한 "man"페지를 보여 준다. man 지령은 다른 지령들에도 적용할수 있으며 체계호출, 서고루틴들, 기타 정보에도 적용할수 있다. 또 다음의 실례에서 보여 주는바와 같이 체계로부터 man 지령 그자체에 대한 정보를 얻을수 있다. 다음의 실례는 우와 같은 체계에서 실행시킬수 있는것이다.

martyp $ **man man**

Reformatting page. Wait ... done

사용자지령 man(1)

이름

man – 참고안내페지를 찾고 현시한다.

표준일치

man [-] [-adFlrt] [-M path] [-T macro - package] [-s section] name ...

man [-M path] -k keyword ...

man [-M path] -f file ...

해설

지령 man은 참고안내서의 정보를 현시한다. 이 지령은 사용자가 선택한 이름에 대한 완전한 안내페지 혹은 개요를 현시한다. 사용자는 관련파일(-f)의 이름을 선택할수도 있다. 안내페지가 없으면 man은 오유통보를 내보낸다.

원천형식

참고안내페지들에는 nroff(1) 혹은 sgml(5)태그(sgml-Standard Genearalized Markup Language: 표준일반표식언어)로서 표식이 붙는다. 지령 man은 이러한 표식들의 형태를 인식하고 그에 따라 파일들을 처리한다. 여러가지 원천파일들이 표식의 형태에 의존하여 분리된 등록부들에 보관된다.

안내페지위치

직결참고안내페지들은 보통 /usr/share/man에 들어 있다. nroff 원천들은 /usr/share/man /man*등록부들에 들어 있다. SGML 원천들은 /usr/share/man/sman*등록부들에 들어 있다. 매개 등록부는 지령소개의 한개 구획에 대응된다. 이러한 등록부들은 선택적으로 설치되기때문에 자기의 주콤퓨터에 남아 있지 않을수 있다. 이 경우에 사용자는 /usr/share/man을 주콤퓨터에 설치해야 한다.

만일 이미 형식화되여 있다면 man은 대응하는 cat * 혹은 fmt *등록부들안에 있는 최신판본들을 단순히 현시하거나 그 판본들을 인쇄한다. 만일 이미 형식화된 요구하는 판본이 날자가 틀리거나 없으면 man 은 그것을 현시할수 있게 재형식화하고 cat * 혹은 fmt *이 쓰기가능하면 재형식화된 판본을 기억시킨다.

--More--（13%）

우의 실례는 man 지령에 대한 "man"페지의 13%만을 보여 준다. "표준일치"아래에서 "man"페지의 구획번호(Section Number)를 규정한다. "man"페지는 몇개의 부류로 그룹화된다. 보통 이 그룹(구획)은 8개 정도이다. 사용자는 어떤 지령에 대한 구획번호를 알고 있으면 그 구획번호를 규정하면 된다.

다음의 실례에서는 passwd 지령에 대한 정보가 서로 다른 몇개의 구획으로 나타나는 것을 볼수 있다. 실례에서는 "man"페지의 구획 1과 구획 4를 보여 준다.

martyp $ **pwd**
/usr/man
martyp $ **find . -name pasewd***
./smani/passwd.1
./sman4/passwd.4
martyp $ **man -s 1 pasowd**
Reformatting page. Wait ... done

```
사용자지령                                                    passwd(l)

이름
   passwd - 가입통과암호와 통과암호속성을 변경시킨다.

표준일치
   passwd [-r | files | -r nis | -r nisplus ] [name ]
   passwd [-r files ] [-egh ] [ name ]
   passwd [-r files ] -s [ -a ]
   passwd [-r files ] -s [ name ]
   passwd [-r files] [-d | -1 ] [ -f ] [-n min ] [-w warn ] [ -x max ] [ name ]
   passwd -r nis [ -egh ] [ name ]
   passwd -r nisplus [-egh] [-D domainname ] [name ]
--More--(3%)

martyp $ man -s 4 passwd
Reformatting page. Wait ... done

파일형식                                                      passwd(4)

이름
   passwd - 통과암호파일

표준일치
   /etc/passwd

해설
   /etc/passwd 는 사용자들과 가입자들에 대한 정보의 국부원천이다. 통과암호파일은
   passwd.byname 및 passwd.bygid 를 넘기는 NIS 와 NIS+ 표 passwd 를 포함하여 다른 통
   과암호원천들과 함께 리용될수 있다. 프로그람들은 이 정보에 접근하기 위하여
   getpwnam(3C)루틴을 리용한다.

   매개 passwd 항목은 다음과 같은 형식의 한개 행이다.

   username:password :uid:gid:gcos-field:home-dir:login-shell
   where
--More--(13%)
```

이 실례에서 리용한 지령들에 대한 해설을 아직 하지 않았지만 그중에서 passwd.1 과 passwd.4 는 지금 시점에서도 흥미가 있는것이다.

passwd에 대한 "man"페지의 구획 1은 앞의 실례와 동일하며 구획 4는 /etc/passwd 파일이다. 이것들은 서로 다른 "man"페지들이다. 즉 하나는 passwd 지령에 대한것이고 다른 하나는 /etc/passwd 파일에 대한것이다. 이렇게 하나의 실마리어에 대한 여러개의 "man"페지가 존재한다. 표준적으로 구획번호를 특별히 규정하지 않는 한 "man"페지의 첫 구획이 나타난다.

Linux와 같은 다른 UNIX 변종들은 구획번호지적에 대문자 "S"를 리용한다. 우의 실례에서는 소문자 "s"를 리용하고 있다. 이 책에서는 여러 UNIX 변종들에서 나타나는 지령사용법의 차이점들을 많이 포함하고 있다. 그러나 이러한 차이점들을 념두에 두면서 자기 체계의 "man"페지를 참고하면 된다.

passwd 지령에 대한 "man"페지를 찾기 위하여 find 지령을 쓰지 않고 man 지령의 "-k" 선택항목을 쓸수 있다. 선택항목 "-k"는 Keyword 단어로부터 나온것이다. 이 선택항목과 함께 passwd에 대한 "man"페지찾기를 하면 passwd가 나타날 때마다 직결개요가 생성된다. 다음의 실례는 이러한 실마리어의 찾기에 의하여 생성된 직결개요들을 보여 준다.

```
# man -k passwd
d_passwd          d_passwd (4)- dial-up password file
getpw             getpw (3c)- get passwd entry from UID
nispasswd         nispasswd. (1)- change NIS+ password information
nispasswdd        rpc.nispasswdd (1m) - NIS+ password update daemon
passwd            passwd (1)- change login password and password attributes
passwd            passwd (4)- password file
pwconv            pwconv (1m)- installs and updates /etc/shadow
                  with information from /etc/passwd
rpc.nispasswdd    rpc.nispasswdd (1m) - NIS+ password update daemon
rpc.yppasswdd     rpc.yppasswdd (1m) - server for modifying NIS password file
yppasswd          yppasswd(1)- change your network password in the NIS database
yppasswdd         rpc.yppasswdd (1m) - server for modifying NIS password file
```

이 지령은 find지령이 만들어 내는 passwd에 대한 긴 "man"페지목록을 자료기지로부터 생성한다. 여기에는 passwd에 대한 구획 1, 구획 4도 포함된다. "-k"선택항목을 리용하려면 체계관리자가 catman지령으로 "man"페지자료기지를 만들어야 한다.

전자우편

모든 UNIX 변종들은 전자우편프로그람으로 호상 전달된다. 자기의 체계에서 전자우편프로그람을 리용하여 동일한 체계를 가지는 다른 사용자들에게 전자우편물을 보내거나 그 사용자들로부터 받을수 있다. 또한 다른 체계의 사용자들과도 통신할수 있으며 체계

관리자가 인터네트설치를 하고 있다면 인터네트와도 통신할수 있다. 이 장에서는 전자우편을 주고받는 기초개념을 서술한다.

체계에 처음으로 가입한후에 사용자는 전자우편을 받을수 있으며 통과암호를 변경시키라는 요청을 받을수도 있다. 모든 UNIX 변종들에 있는 전자우편프로그람은 mail 이다. mail 은 대단히 사용하기 쉬운 지령이다. 다음의 실례에서는 mail 프로그람을 가동시키고 mail 지령들의 목록을 출력하도록 지령 "?"을 실행시키며 Tom 으로부터 통보를 읽은 다음 거기에 대답하며 초기의 통보를 보관한다.

```
* Welcome to sys1
Last unsuccessful login: Mon Mar 8 09:32:13
Last login: Mon Sep 27 07:21:01

*******************************************************
* This is a private system operated for use by our company only.      *
*******************************************************

* Welcome to sys1
You have mail.
TERM set to vt100

sys1:/home/martyp
martyp $ mail
Mail [5.2 UCB] Type ? for help.
"/var/spool/mail/martyp": 1 message 1 new
>N 1 tomf              Mon Sep   27   07:23   11/392   " 12 : 00  Phone  Call"
? ?
Control Commands:
   q                   Quit - apply mailbox commands entered this session.
   x                   Quit - restore mailbox to original state.
   ! <cmd>             Start a shell,  run <cmd>,  and return to mailbox.
   cd [<dir>]          Change directory to <dir> or $HOME.
Display Commands:
   t [<msg_list>]      Display messages in <msg_list> or current message.
   n                   Display next message.
   f [<msg_list>]      Display headings of messages.
   h [<num>]           Display headings of group containing message <num>.
Message Handling:
   e [<num>]           Edit message <num> (default editor is ex).
   d [<msg_list>]      Delete messages in <msg_list> or current message.
```

u [<msg_list>]　Recall deleted messages.
s [<msg_list>] <file>　Append messages (with headings) to <file>.
w [<msg_list>] <file>　Append messages (text only) to <file>.
pre [<msg_list>]　Keep messages in system mailbox.

Creating New Mail:

m <addrlist>　Create/send new message to addresses in <addrlist>.
r [<msg_list>]　Send reply to senders and recipients of messages.
R [<msg_list>]　Send reply only to senders of messages.
a　Display list of aliases and their addresses.

========================= Mailbox Commands =======================

? **t**
Message 1:
From tomf Mon Sep 27 07:23:30
Date: Mon, 27 Sep 1999 07:23:30 -0700
From: <tomf>
To: martyp
Subject: 12:00 Phone Call

Please call me at 12:00 CA time to discuss trip.
Tom

? **R**
To: tomf
Subject: Re: 12:00 Phone Call

I'll call you then.
Marty
Cc:
? **s**
"/home/martyp/mbox" [Appended] 11/392
? **q**
sys1:/home/martyp
martyp $

mail 이라고 입력하면 전자우편프로그람이 기동된다. 그러면 12:00 에 전화호출이 있었다는 통보를 tomf 로부터 받는다. 그 다음에는 ?를 입력하여 mail 지령목록을 얻는다. 그다음 t 를 입력하면 받은 통보를 현시하여 주며 R 를 입력하면 통보를 보낸 tomf 에게 대답을 하게 된다. s 지령을 주면 통보를 보관한 다음 지워 버리고 q 를 입력하면 탈퇴한다. mbox 파일이 만들어 졌는데 그것은 보관통보를 모두 포함한다.

이 실례를 통하여 우편함의 통보를 어떻게 조종하는가를 보았다. 즉 보려는 통보를 요약하여 보고 보관한 다음 제거하였다. mbox 는 보관시키는 정보들을 기억시키는 파일이다.

다음의 실례는 mbox를 입력파일로 하여 "-f"선택항목으로 요약된 정보를 보도록 한다.

```
martyp $ mail -f mbox
Mail [5.2 UCB] Type ? for help.
"mbox": 4 messages
 1 tomf          Sun Sep 26 18:14    11/343  "Training"
  >2 donnak      Mon Sep 27 07:23    12/402  "Performance"
 3 carollync     Mon Sep 27 07:27    12/380  "Software Update"
 4 tomf          Mon Sep 27 07:38    11/357  "12:00 Phone Call"
```

이 실례에서 mbox 에 보관된 4 개의 통보를 볼수 있는데 이 통보들은 파일번호(1−4)로 구별되며 사용자는 그것을 가지고 작업할수 있다. 현재 통보는 >기호로 지적되며 매 통보는 통보배달시간과 날자, 통보의 크기(행개수 및 기호개수), 통보의 제목 등의 정보로 되여 있다. s 로 파일을 보관하고 제거하는것은 후에 그 파일에 접근할수 있도록 담보하여 준다.

보다 발전된 전자우편프로그람은 일부 체계들에서 mailx 라고 부르는데 mailx 는 통보목록을 현시할뿐아니라 통보목록을 생성하기도 한다. mailx 통보목록은 방금 본 mail -f mbox 를 제거하고 보관시킨 목록과 류사하다. mailx 에 의하여 자기가 통보를 읽었는가 읽지 않았는가를 알수 있다. 매개 통보의 앞에는 N 이나 O 라는 문자가 표시된다. N 은 새로운것(New), O 는 한번 읽은 낡은 통보(Old)를 의미한다. 그리고 U 는 읽지 않은(Unread)통보라는것을 나타낸다. Red Hat Linux 와 같은 일부 체계들에서는 mail 을 실행시켜도 mailx 와 같은 통보목록을 얻을수 있다.

다음의 실례는 mailx 를 수행시킬 때 무엇이 나타나는가를 보여 준다.

```
martyp $ mailx
Mail [5.2 UCB] Type ? for help.
"/var/spool/mail/martyp": 2 messages 1 new 2 unread
  U 1   tomf              Mon Sep 29 08:40   13/394  "Networking"
>N 2   donnak     Mon Sep 29 08:49   10/342  "Interoperability"
?
```

이 실례에서 첫 통보는 읽지 않은 U 이고 둘째 통보는 새로운 통보 즉 N 이다. 기호 ">"는 새로운 통보에 있다. 통보보관은 mbox 파일에만 하는것이 아니라 다른 파일에도 할수 있다.

다음의 실례는 trip 파일에 통보를 보관시키는것을 보여 준다.

```
martyp $ mailx
Mail [5.2 UCB] [AIX 4.1]Type ? for help.
"/var/spool/mail/martyp": 2 messages 2 unread
>U 1   tomf          Mon Sep 27 08:49 11/352 "NY trip"
 U 2  donnak         Wed Sep 29 04:44 11/354 "login change"
?
Message  1:
From tomf Mon Sep 27 08:49:09
Date: Mon,  27 Sep 08:49:08 -0700
From: <tomf>
To: martyp
Subject:   NY trip

I am going to change the date of the NY trip.

? s trip
"trip" [New file] 11/352
? q
Held 1 message in /var/spool/mail/martyp
martyp $
```

trip 파일은 s trip 로서 통보를 보관할 때 생성되였다. 이제부터는 이 파일에 모든 관련통보들을 기억시킬수 있고 파일에 보관된 통보는 lp 지령으로 인쇄할수 있다. lp 지령은 후에 구체적으로 보겠지만 지금은 lp filename 으로 표준인쇄기에 내보내도록 파일만 지적하면 된다. trip 파일을 인쇄하려면 lp trip 라는 지령을 주면 된다.

이처럼 전자우편을 보내는 일은 대단히 쉽다. 즉 mailx 나 mail 을 입력한 다음 통보를 보내려는 사용자의 이름만 주면 된다. tomf 에게 보내는 경우 다음과 같이 한다.

```
martyp $ mail tomf
Subject: Pickup
I'll be at the airport in NY to give you a ride.
Marty
Cc:
martyp $
```

^D 가 전자통보의 끝에서 리용되였다. 이 통보는 tomf 의 우편함에 들어 간다. 통보는 또한 인터네트이름으로도 보낼수 있다. 다음의 실례는 tomf 의 인터네트우편주소로 통보를 보내고 있다.

martyp $ **mail tomf@tomcompany.com**
Subject: Pickup
I'll be at the airport in NY to give you a ride.
Marty
Cc:
martyp $

통보를 다른 체계나 인터네트주소로 보내려면 체계관리자가 이 기능을 설치하여야 한다. 도형대면부를 가진 체계에도 우편물을 관리하는 수단이 있다. 비록 그것이 도형환경에서 수행된다고 하여도 그것들은 모두 앞에서 서술한 우편프로그람들을 리용한것이다.

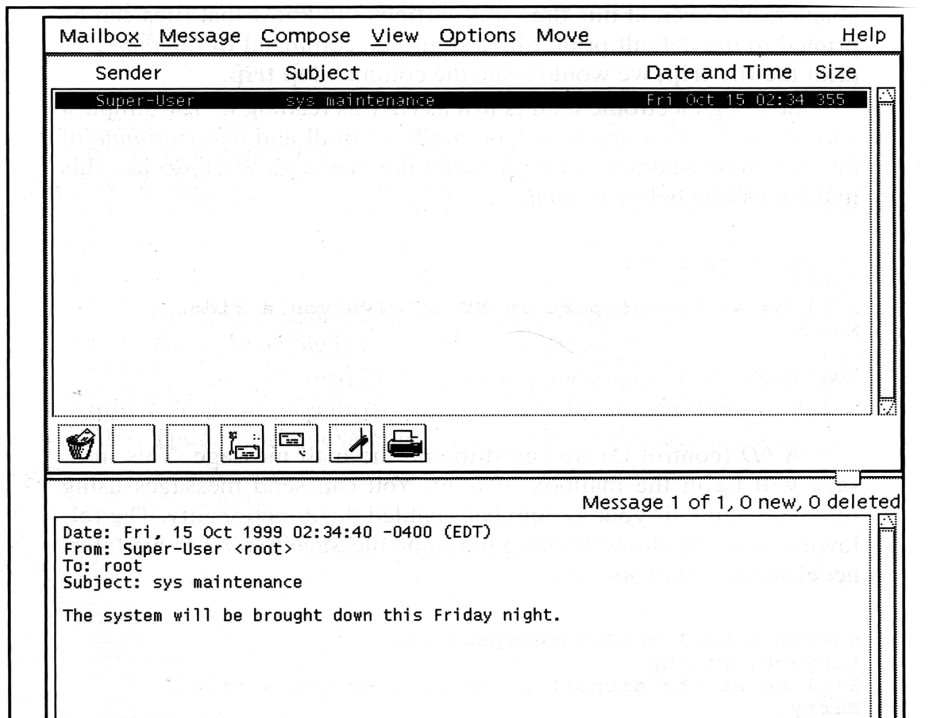



그림 1-2. 공동탁상환경에서 전자우편의 접근방법

그림 1-2 의 꼭대기에 있는 내리펼침차림표에서는 우리가 이미 학습한 보관, 전진, 제거 등의 지령들을 선택하여 실행시키도록 한다.

인터네트에 접근

인터네트는 그 시초부터 UNIX 체계들에서 광범히 리용되여 왔다. 맨 처음으로 많이 리용된 **Web 열람기**(Web Browser)는 일리노이즈대학에서 개발된 Mosaic 라는 X Window 체계의 응용프로그람이다. 현재도 Mosaic, Netscape Communicator 등을 포함하여 많은 Web

열람기가 UNIX 체계들에서 리용되고 있다. 여기서는 UNIX 체계들에서 가장 광범히 리용되고 있는 Netscape Communicator 에 대하여 고찰한다. 이것의 계렬제품들이 많이 사용되고 있는데 그가운데서 Netscape Navigator 가 기본적인 Web 열람기이며 다른것들도 동일한 원리로 적용할수 있다.

Linux Red Hat 와 같은 여러 UNIX 변종들은 배포판에서 이미 열람기들을 제공해 주고 있다. 열람기와 인터네트는 UNIX 체계에서 리용되지만 UNIX 체계와는 별도의 개별적인 도구들이다. 그림 1-3 은 Red Hat 의 홈페지를 현시하는 Netscape Communicator 창문을 보여 준다.

그림 1-3. 홈페지에 접근하기 위한 열람기의 사용방법

이 홈페지에서 Red Hat Linux 조작체계와 관련되는 임의의 문제에 대한 탐색을 할수 있다. 홈페지에 대한 접근은 열람기의 **위치칸**(Location Box)에 **유일자원지시자**(Uniform Resource Locator)를 입력하면 된다.

Red Hat 의 URL 은 www.redhat.com 이다. UNIX 변종들에서 제일 많이 리용되는 회사들의 URL 을 표 1-1 에서 보여 주었다.

이 홈페지들로부터 필요한 정보를 찾기 위한 작업을 할수 있다. 홈페지에 대한 탐색을 통하여 보려는 Web 싸이트를 재빨리 찾을수 있으며 관련항목들을 리용하여 필요한 관련정보들을 볼수도 있다.

표 1-1 대표적인 UNIX 관련회사의 URL 들

URL	결 과
www.hp.com	Hewlett-Packard 회사
www.ibm.com	IBM 회사
www.redhat.com	Red Hat 회사
www.Sun.com	Sun 마이크로체계회사

그림 1-3 에서 www.redhat.com 앞에 http://가 붙어 있는것을 볼수 있다. 이것은 URL 의 **규약**(Protocol)을 지정한것이다. http://은 Web 페지를 규정할 때 쓰이는 규약이다. 표 1-2 는 가장 많이 리용하는 규약들을 보여 준다.

표 1-2 많이 리용하는 규약들

URL	결 과
file:<pathname>	<pathname>에서 규정한 국부파일을 열람기에 적재한다.
ftp://	파일전송규약이 리용된다.
gopher://	Gopher 봉사를 제공하는 호스트(주콤퓨터)가 리용된다.
http://	목적하는 Web 페지가 리용된다.
https://	목적하는 보호 Web 페지가 리용된다.
news://	망뉴스봉사기가 리용된다.

다음에 나타나는 몇개의 그림들은 열람기에서 수행시킬수 있는 기본적인 전용화면들이다. 이 화면들을 통하여 대부분의 열람기에서 볼수 있는 전용화의 형태들을 상상할수 있다. 이미 열람기가 적재되여 있다고 하여도 자기가 요구하는 형식으로 환경설정을 할수 있다. 그림 1-4 에서는 Netscape Communicator 의 Fonts 에 대한 전용설정을 하고 있다.

이 실례에서는 Appearance 아래에 있는 Fonts 를 선택하여 응용프로그람에 나타나는 서체의 크기를 규정한다. Appearance 아래에는 또한 Colors 가 있는데 이것은 응용프로그람에서의 색을 조절하는 항목들을 포함한다.

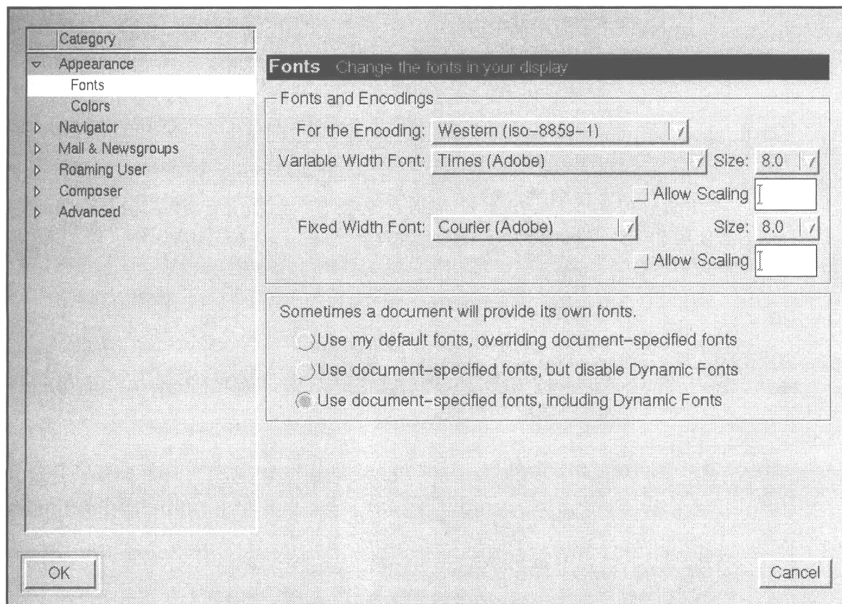



그림 1-4. 열람기의 서체규정방법

그림 1-5 는 Navigator 에서 수정할수 있는 항목들을 보여 준다. Navigator 에는 3 개의 전용항목들이 있는데 현재 그림은 Applications 항목에 대한 창문으로서 응용프로그람들과 그것의 《방조자》들을 표시하여 준다.

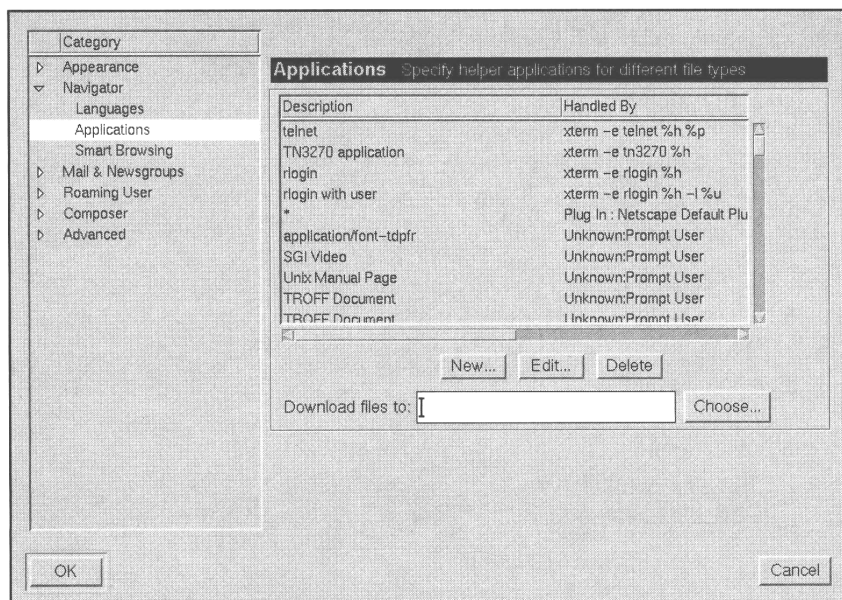



그림 1-5. 열람기의 응용프로그람들

그림 1-6 에서는 Mail & Newsgroups 부분을 보여 주고 있다.

그림 1-6 에서는 통보를 저축할수 있는 디스크공간의 크기를 규정하고 있다. 우편

물이나 새소식묶음들이 대단히 큰 통보량으로 자기 체계에 송신되므로 받는 통보의 량과 시간에 따라 어느 통보를 제거하겠는가를 조종할 필요가 있다.

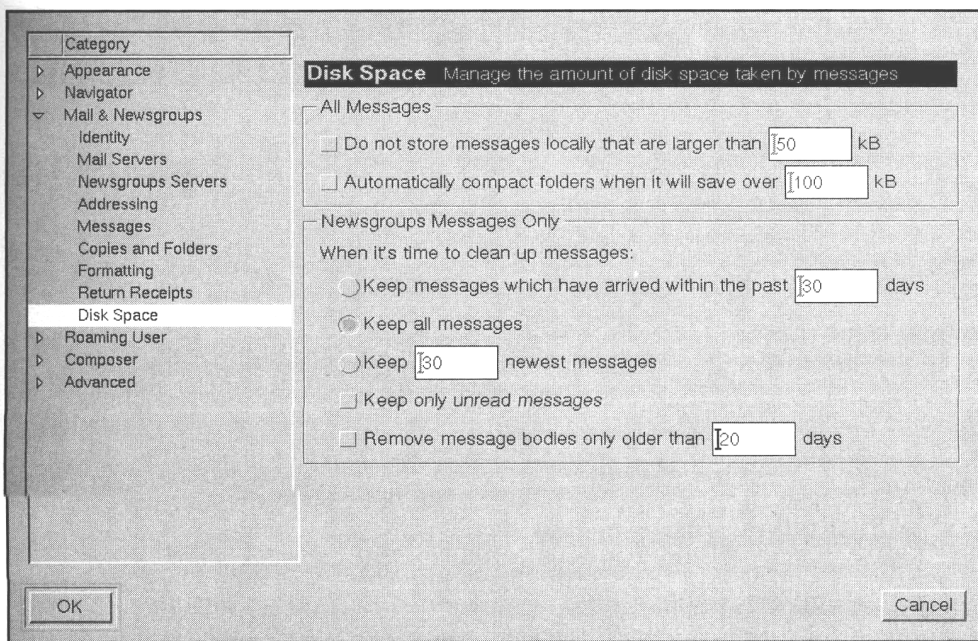



그림 1-6. 열람기의 디스크공간

그림 1-7에서는 다음부류인 **류동사용자**(Roaming User)를 보여 준다.

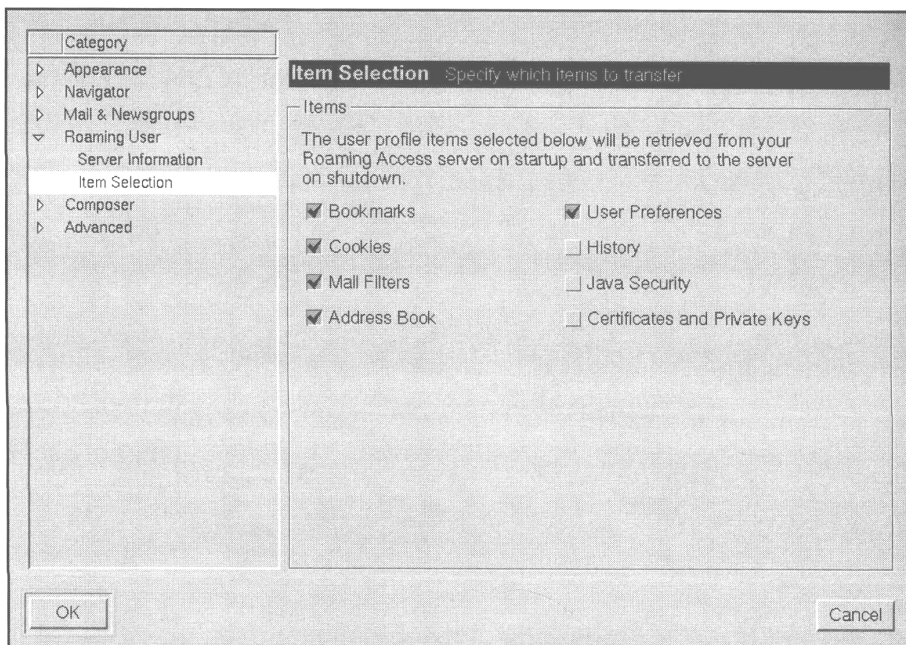



그림 1-7. 류동사용자항목들

그림 1-7 은 사용자가 **류동접근봉사기**(Roaming Access Sever)를 리용할 때 전송하게 될 관리파일들을 보여 준다. 이러한 선택항목들은 어디에서 Netscape 를 리용하든지 관계없이 사용자가 요구하는 형식으로 설정할수 있게 해준다.

다음의 부류 Composing 은 출판하는 Web 문서들에 대한 선택항목을 제공한다(그림 1-8).

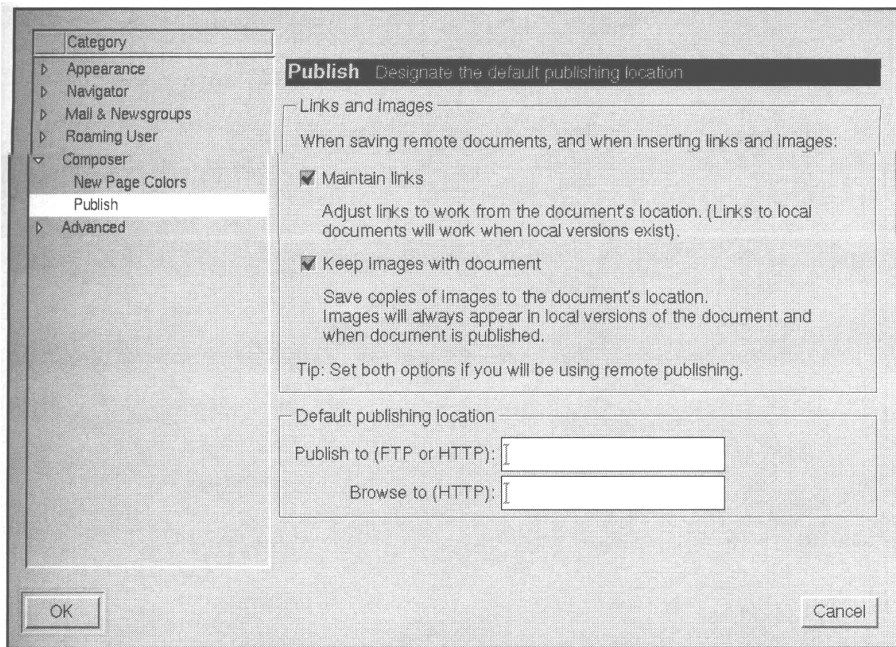



그림 1-8. 출판과 관련된 선택항목

그림 1-8 에서는 Composing 아래에 Publish 를 보여 주는데 여기서는 문서의 위치와 같은 Web 와 관련된 출판항목들을 보여 준다.

마지막부류 Advanced 에는 그림 1-9 에서처럼 Cache 와 Proxies 에 대한 정보를 포함하고 있다.

콤퓨터체계에 대한 침해를 막기 위하여 대부분의 회사들은 체계에 대한 접근을 제한할 목적으로 **방화벽**(Firewall)을 설치하였다. 이 제한은 회사망의 입력 및 출력을 어렵게 하도록 한다는것을 의미한다. **대리인**(Proxy)은 인터네트에 대한 접근을 할수 있게 하는 수단이다.

Category
- Appearance
- Navigator
- Mail & Newsgroups
- Roaming User
- Composer
- Advanced
  - Cache
  - Proxies

Proxies Configure proxies to access the Internet

A network proxy is used to provide additional security between your computer and the Internet (usually along with a firewall) and/or to increase performance between networks by reducing redundant traffic via caching. Your system administrator can provide you with proper proxy settings.

Direct connection to the internet

Manual proxy configuration View...

Automatic proxy configuration

Configuration location (URL)

Reload

OK Cancel

그림 1-9. 대리인의 선택

그림 1-10은 그림 1-9에서 선택한 Proxies항목의 대화칸을 보여 주고 있다.

You may configure a proxy and port number for each of the internet protocols that Netscape supports.

FTP Proxy:	192.168.10.2	Port:	8080
Gopher Proxy:		Port:	
HTTP Proxy:	192.168.10.2	Port:	8080
Security Proxy:		Port:	
WAIS Proxy:		Port:	

You may provide a list of domains that Netscape should access directly, rather than via the proxy:

No Proxy for:

SOCKS Host: Port: 1080

OK Cancel

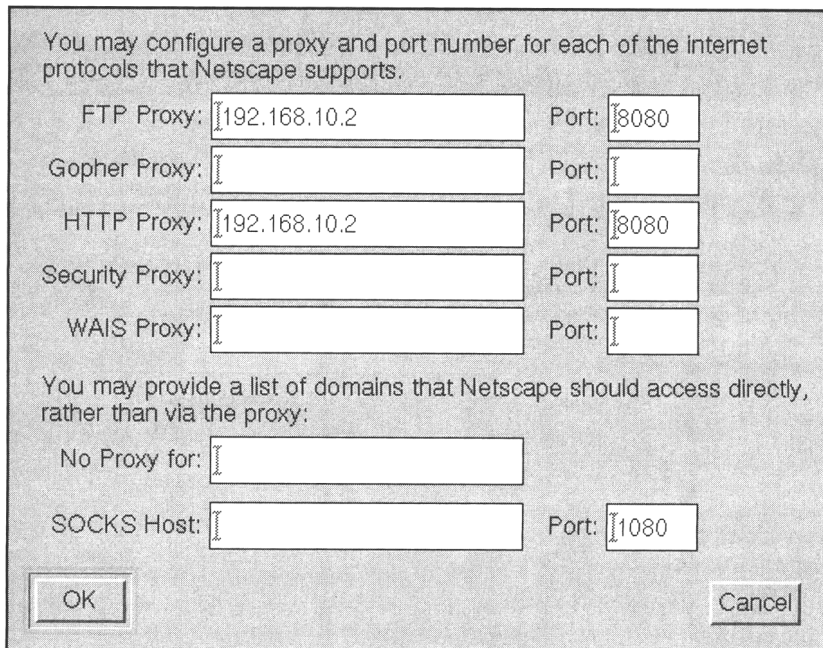



그림 1-10. 대리인과 관련된 정보를 규정

그림 1-10 에서는 인터네트접근을 위한 두가지 형태의 대리인의 주소와 포구번호를 설정하였다. 이 설정은 체계관리자에 의하여 이미 되여 있을수도 있다. 만일 설정이 진행되지 않았다면 이 정보에 대하여 잘 모르는 경우에 이전의 설정내용을 그대로 주도록 요구할수 있다.

그림 1-11 은 Advanced 아래에 있는 Cache 항목을 보여 주고 있다.

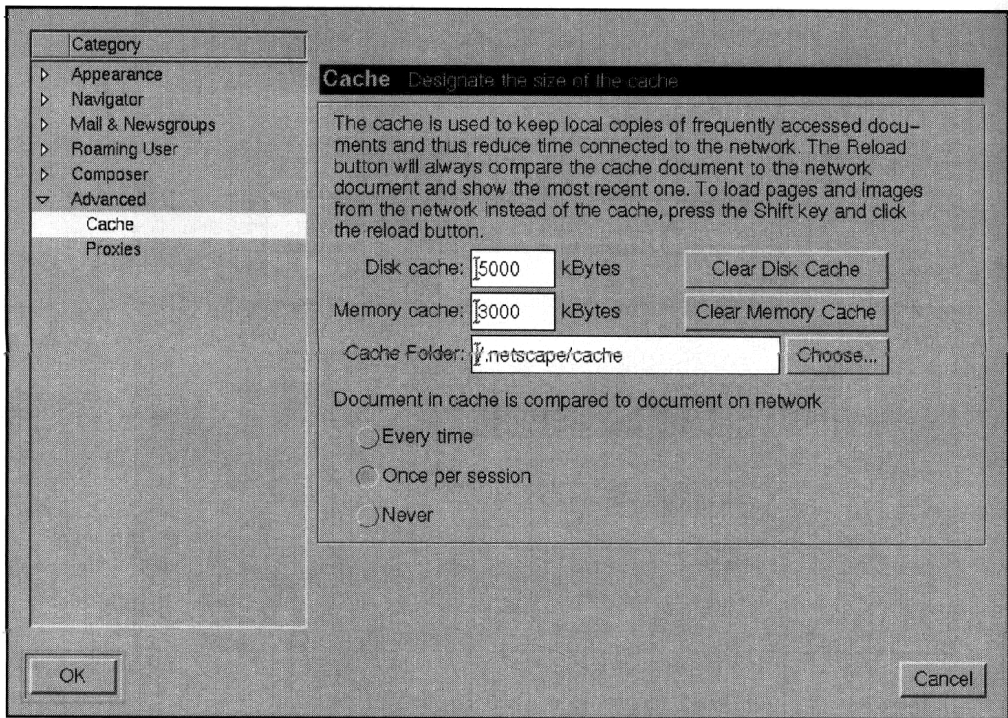



그림 1-11. 고속완충기의 전용화

열람기관련의 **고속완충기**(Cache)는 국부적으로 자주 리용되는 정보를 복사하여 보관시킴으로써 인터네트작업의 속도를 높이는데 리용된다. 결국 정보를 반복적재하려면 방에서 벗어나지 말아야 한다. 이 창문에서는 고속완충기의 크기 및 고속완충되여 보관시킬 자료의 위치를 규정할수 있다.

대부분의 열람기들은 우에서 고찰한 전용화면들을 제공해 준다.

UNIX 구성요소

지금까지 가입후 몇개의 지령도 써보았고 인터네트에도 접근하였으므로 그 과정에 UNIX 체계와 호상작용하는 방법을 알게 되였다고 볼수 있다. 따라서 이제는 UNIX 체계를 구성하고 있는 높은 수준의 구성요소들을 취급할 때가 되였다.

그림 1-12는 UNIX체계를 추상적으로 묘사한것이다.

UNIX 체계의 구성요소들을 보는 견해는 각이하다. 그림 1-12 는 사람들의 견해에서 제일 보편적이라고 말할수 있는 구성요소들을 제시하고 있다.

그림의 제일 중심에 있는 원은 **장치체계**(Hardware)를 표시한다. UNIX 는 여러가지 형태의 장치에서 동작한다. UNIX 와 관련되는 장치체계의 종류는 너무 많기때문에 여기서는 장치체계에 대한 서술을 많이 하지 않기로 한다.

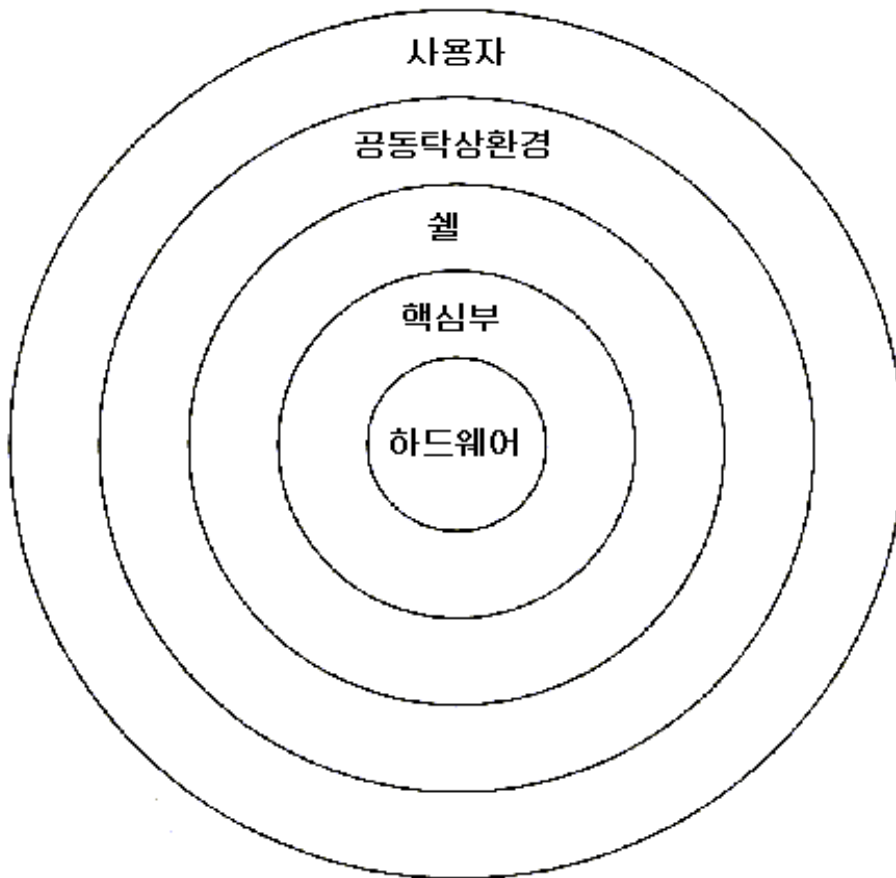



그림 1-12. 높은 준위의 UNIX체계구조

그 다음의 원은 **핵심부**(Kernel)를 표시한다. 핵심부는 장치관리, 기억관리, 과제관리와 모든 지령의 계획화와 실행 그리고 장치를 조종하는 구동프로그람의 관리 등을 비롯하여 많은 기능을 수행한다. 핵심부는 체계관리자가 대부분 유지관리하는 체

계의 기본부분이다. 새로운 장치가 하나 추가되면 장치구동프로그람이 핵심부에 첨부되며 핵심부가 그 새로운 장치를 관리하여야 한다. 또한 UNIX 체계에서 동작하는 응용프로그람들에 대하여 체계의 성능을 최량화하도록 핵심부로 동작할수 있는 실제적인 조종부가 있다. 사용자들은 지령을 통하여 간접적으로만 핵심부부분과 호상 작용할수 있다.

그 다음의 원은 지령처리기 **쉘**(Shell)을 표시한다. 쉘은 사용자로부터 지령들을 받아 그것을 수행시킨다. 이 부분은 사용자들이 체계에 표준적으로 접근하는 부분이다. 쉘은 지령들을 보다 깊은 층(안쪽 원)으로 전달하는것외에도 지령들을 배경에서 수행시키고 쉘프로그람을 실행시킬수 있게 한다. 이 책에서는 제일 공통적인 3 개의 쉘과 기본적인 쉘프로그람작성법에 대하여 고찰하게 된다.

그 다음원은 공동탁상환경(CDE) 혹은 다른 도형대면부를 표시한다. 제 4 장에서는 CDE 의 많은 실례를 제시하게 되며 Linux 체계에서 동작하는 Gnome 라는 또 다른 도형대면부에 대하여 고찰하게 된다. 보통 이러한 UNIX 도표에서는 도형사용자대면부를 하나의 분리된 원으로 고찰하지 않는다. 왜냐하면 도형대면부에서 수행하는 대부분의 작업이 내부적으로는 쉘지령에 의하여 수행되기때문이다.

마지막원은 사용자의 원을 표시한다. 이 그림에서 보여 준 모든 층들을 통하여 사용자는 UNIX 체계에서의 작업을 수행할수 있다.

이 책에서 주는 많은 실례가 지령행으로 수행되며 일부는 도형대면부로 수행된다. 그러므로 사용자들은 자기가 실지로 어떻게 UNIX 의 내부원과 련결되는지는 모를수 있다. 체계관리자, 자료기지관리자, 기타 체계유지성원들이 이러한 내부구성요소들을 고찰하므로 그림 1-12 는 이러한 성원들이 작업하는 령역의 진가에 대하여 알수 있게 한다. 쉘, 우편체계, 사용자대면부, 열람기대면부, 기타 구성요소들은 전용화될수 있는데 그 내용은 다음장들에서 UNIX 파일체계와 지령들을 리용하면서 고찰하게 된다.

제2장. UNIX 파일체계의 소개 - 파일체계구조, file 및 ls 지령

UNIX 에서의 모든 작업은 파일과 등록부를 통하여 진행된다. 그러므로 UNIX 에서의 파일 및 등록부관리방법과 파일체계의 계층구조를 잘 알면 UNIX 를 충분히 리해할수 있다.

이 장에서는 파일 및 등록부와 관련하여 제일 많이 리용되는 file 지령과 ls 지령에 대하여 고찰하기로 한다. file 은 파일의 종류를 결정하며 ls 는 파일 및 등록부를 현시하는데 리용된다.

많은 작업이 UNIX 파일과 등록부에서 진행되므로 이 장에서는 다음의 지령들을 포함하여 파일 및 등록부의 기본내용과 배경에 대하여 서술한다.

- file 지령
- ls 지령
- 파일체계구조

여기서 서술하는 지령들은 모든 UNIX 변종들에서 사용할수 있다.

파일종류

UNIX 체계에서 정보를 기억시키는 모든 수단을 **파일**(File)이라고 부른다. 즉 사용자가 입력하는 지령들, 사용자가 사용하는 응용프로그람들과 자료들, 인쇄기나 건반과 같은 접속장치들까지 모두 파일에 포함된다. 이것은 UNIX 를 단순하게 하기도 하고 복잡하게 하기도 한다. 그 단순성은 UNIX 체계에서 리용하는 모든 대상에 대하여 해당한 파일이 존재한다는것을 알면 된다는데 있으며 복잡성은 파일의 내용이 ASCII 본문파일로부터 실행프로그람까지 임의의 령역이라는데 있다.

체계에서의 모든 파일은 **파일이름**(Filename)을 가진다. 조작체계는 모든 파일체계관련의 과제를 관리하는데 사용자는 파일이름과 그것의 리용방법을 알 필요가 있다. UNIX 체계에는 여러가지 파일형태들이 있다. 일부 파일형태는 리용하는 UNIX 변종에 고유한 파일형태로 된다. 실례로 장치파일은 어떤 UNIX 변종을 수행시키는 특수한 장치의 **가동환경**(Platform)에 대한 정보를 포함한다. 그러나 일반적으로 대부분의 파일종류는 모든 체계들에서 류사하다.

여기서 고찰하는 **파일종류**(Filetype)는 다음과 같은것들이다.

- 본문파일
- 자료파일
- 원천코드파일
- 실행파일

- 쉘프로그람
- 련결
- 장치파일

이러한 매개 종류의 파일은 종류에 따르는 **확장자**(Extension)를 가진다. 많은 응용프로그람은 자기의 고유한 확장자를 가지고 있다. 본문파일은 .txt 확장자를 가지며 C 원천코드프로그람은 .c 확장자, 대부분의 C++프로그람은 .cc 확장자를 가진다. 확장자는 파일종류를 결정하는데는 효과적이지만 어떤 파일종류가 반드시 규정된 확장자를 가져야 한다거나 또 무조건 확장자가 있어야 한다는 요구는 전혀 없다.

다음의 표는 공통적으로 리용되는 여러가지 파일종류의 확장자들을 보여 준다.

확장자	파 일 종 류
.a	체계관리 혹은 서고파일
.c	C 프로그람원천
.cc	C++프로그람원천
.csh	C 쉘스크립트(script)
.f	FORTRAN 프로그람원천
.ksh	Korn 쉘스크립트
.o	콤파일된 원천의 객체파일
.ps	포스트스크립트원천
.l to .n	직결안내원천
.shar	쉘관리파일
.sh	Bourne 쉘스크립트
.tar	tar 체계관리파일
.txt	ASCII 본문파일
.Z	압축파일

파일종류를 결정하는 효과적인 수법은 file 지령을 리용하는것이다. UNIX 변종들은 출력형식이 좀 차이나도 거의나 file 지령을 제공해 주므로 이 지령에 의하여 파일종류를 결정하는것이 좋다.

본문파일

ASCII 기호들은 사용자가 수행하는 작업을 표현하는 글자들과 수자들이다. 실례로 전자우편통보나 편지를 작성하는 UNIX 편집기는 **본문파일**(Text File)을 만든다. 아래에서는 Linux 체계의 ASCII 본문파일의 일부분을 보여 준다.

Thu Nov 5 07:29:53 1999 debug: FTS: appending '/dev/hdal'
Thu Nov 5 07:29:53 1999 debug: FTS: appending '/tmp/LST/swap.sel'
Thu Nov 5 07:29:53 1999 debug: Building partition-list with argument <Linux>
Thu Nov 5 07:29:53 1999 debug: Respecting exclude file /tmp/LST/targets.sel:
/dev/hdd
/dev/hda1
/dev/hda2
Thu Nov 5 07:29:53 1999 debug: No ADDITIONAL partitions available
Thu Nov 5 07:29:59 1999 debug: added '83901 to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug: added '8390' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07 29 59 1999 debug: added 'scsi_mod' to '/root/tmp/modules.handled'
Thu Nov 5 07 29 59 1999 debug: added 'scsi_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: added 'sd_mod' to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug: added 'sd_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:46 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: added 'sr_mod' to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug:added 'sr_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug; added 'st' to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug: added 'st' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: added 'sg' to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug:added 'sr_mod' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: added 'nfs' to '/root/tmp/modules.handled'
Thu Nov 5 07:29:59 1999 debug: added 'nfs' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: added 'isofs' to '/root/tmp/modules.handled,
Thu Nov 5 07:29:59 1999 debug: added 'isofs' to '/root/etc/modules/2.0.29/#1 Tue Feb 11 20:36:48 MET 1997.default'
Thu Nov 5 07:29:59 1999 debug: 'isofs' is already handled
Thu Nov 5 07:29:59 1999 debug: 'nfs' is already handled
Thu Nov 5 07:29:59 1999 debug: 'sg' is already handled
Thu Nov 5 07:29:59 1999 debug: is already handled

자료파일

응용프로그람에서 사용하는 자료들을 포함하는 파일은 **자료파일**(Data File)이다. 책을 만드는 FrameMaker 와 같은 복잡한 탁상출판용도구를 리용하는 경우에 이 도구가 사용하는 자료파일을 창조하게 된다. 사용자는 이러한 자료파일들을 읽을수 있으며 또 정보를 형식화하여 읽기만 하도록 숨겨 놓을수도 있다. UNIX 를 설치할 때에 자료기지프로그람을 리용하는 경우가 있는데 이때 사용하는 자료파일은 부분적으로만 읽을수 있는 파일이다.

원천코드파일

원천코드파일(Source Code File)은 C, C++, Pascal, FORTRAN 과 같은 프로그람작성언어와 관련되는 정보를 포함하는 본문파일이다. 이러한 파일은 적어도 그것을 작성한 프로그람작성자만은 읽을수 있는 파일이다. 프로그람작성자들은 원천코드파일을 개발할 때 프로그람언어에서 리용되는 습관적인 이름으로 파일을 창조한다. 실례로 C 프로그람을 창조하면 파일뒤에 ".c"가 붙는다. 다음의 실례는 C 원천코드파일에 대하여 보여 주고 있다.

```
/* this is K & R sort program*/
# include <stdio.h>
# include <stdlib.h>

    int N;
    int v[1000000];  /* v is array to be sorted*/
    int left = 0;     /* left pointer*/
    int right;
    int swapcount, comparecount = 0;
                      /*count swaps and compares*/
    int i, j, t;
    char print;
    char pr_incr_sorts;
main( )

{
printf("Enter number of numbers to sort : ");
scanf("%10d", &N);        /* 10d used for a BIG input*/
printf ("\n");            /* select type of input to sort*/

printf("Enter rand(1), in-order(2), or reverse order (3)      sort : " );
```

```
scanf("%2d", &type);
printf ("\n");                  /*select type of input to sort*/

if (type = = 3)
    for (i=0; i<N; ++i)         /* random*/
            v[i] = (N-i) ;

else if (type = = 2)
    for (i=0; i<N; ++i)
            v[i]= (i + 1);      /* in order*/

else if (type = = 1)
    for (i=0; i<N; ++i)
            v[i]=rand( );       /* reverse order*/
fflush(stdin);
printf("Do you want to see the numbers before sorting (y or n)?
scanf("%c", &print);
printf ("\n");                  /*View unsorted numbers?*/
if (print == 'y')
  {
    printf ("\n");
    for (i=0; i<N; ++i)
      printf("a[%2d]= %2d\n", i, v[i]);
      printf ("\n")
  }
ffl ush(stdin);
printf ("Do you want to see the array at each step as it sorts? (y or n) ?
scanf("%c", &pr_incr_sorts);
printf ("\n");                  /*View incremental sorts?*/

    right = N-1;                /*right pointer*/
            qsort(v, left, right);

{
    fflush(stdin);
    printf ("Here is the sorted list of %2d items\n", N);
            printf ("\n");
    for (i=0; i<N; ++i)
            printf ("%2d\n 11, v[i]);
```

```
            printf ("\n") ;
            printf ("\n");     /*print sorted list*/
              }
                  printf ("number of swaps = %2d\n 11, swapcount);
                  printf ("number of compares = %2d\n    comparecount);
    }
/* qsort function*/
            void qsort( v, left, right)
                          int v[ ], left, right;
    {
              int i, last;
              if (left > right)
                  return;

              swap(v, left, (left + right)/2);
              last = left;
              for (i=left+1; i <= right; i++)
              {
                  comparecount = ++comparecount;
                  if (v[i] < v[left])
                  swap(v, ++last, i);
              }
              swap(v, left, last);
              qsort(v, left, last-1);
              qsort(v, last+1, right);
              }
              /* swap function*/

                  swap(v, i, j)
                  int. v[ ], i, j;

                  {int temp;
                   swapcount = swapcount++;
                   temp = v[i];
                   v[i] = v[j];
                   v[j] = temp;

if (pr_incr_sorts = = 'y')
 {
```

```
    printf("Incremental sort of array = ") ;
    printf ("\n") ;
    for (i=0; i<N; ++i)
            printf("a[%2d]= %2d\n", i, v[i]);
            printf ("\n");
    }
}
```

실행파일

실행파일(Executable File)은 수행할수 있게 콤파일되거나 해석된 프로그람이다. 실행파일은 읽어 볼수 없으므로 그것을 보려면 오유묶음이나 읽기불가능한 기호들이 나타나면서 UNIX 체계로부터 경고가 울린다. 이때에는 현재의 화면설정을 잃을수 있으며 다른 문제도 산생될수 있다. 실행파일은 UNIX 안으로 많이 들어 가지 않아도 도처에서 찾아 볼수 있다. 사용자가 입력하는 UNIX 지령의 대부분은 읽을수 없는 실행파일이다. 또한 체계에서 프로그람을 개발한다면 실행파일을 창조하는것으로 된다.

아래에서는 실행파일을 화면에서 읽어 보려고 할 때 나타나는 현상을 제시해 주고 있다.

```
unknown/etc/ttytyperunknown<@=>|<@=>|:unknown<@=>
callocLINESCOLUMNSunknownPackaged for
argbad aftger%3
parmnumber missing <@=>|<@=>|:
@ @ 3### @@@A:2TTO|>@#<|2XOOR
EraseKillOOPS<@=>|<@=>|:
<@=>|<@=>|:
<@ =>|<@=>|:<@=>|ATOO<@=>|:<@=>|<@=>|:<@=>|<@=>|:<@=>|<@=>|:
```

쉘프로그람

쉘프로그람(Shell Program)은 과제를 수행하기 위하여 동작시키는 파일이며 읽어 볼수 있는 파일이다. 이 파일은 실행가능하므로 동작시킬수도 있고 읽어 볼수도 있다. 뒤장에서는 쉘프로그람작성법을 보다 자세히 서술한다. 쉘프로그람작성법은 사용자들이 알아야 할 중요한 기교이다. 파일 및 **허락**(Permission)과 관련한 일부 배경은 쉘프로그람작성의 토대로 되는 중요한 내용이므로 사용자들은 파일 및 허락과 관련한 지식을 알아야 한다.

다음의 실례는 체계의 재정계산을 진행하는 쉘프로그람의 부분을 보여 준다.

```
#!/bin/sh
{

echo "PROG>>>>> determining if Ignite-UX loaded on system. "

igtest='swlist    |    grep Ignite'
echo $igtest
if  [ -n "$igtest" ]; then
echo "Ignite-UX installed"
else
echo "Ignite-UX is not installed"
fi

echo "PROG>>>>> determining if make_recovery has been run by checking for
/var/opt/ignite/arch.include."

if [ -f /var/opt/ignite/recovery/arch.include ]; then
echo "make-recovery has been run."
else
echo "make-recovery has not been run."
fi
echo "PROG>>>>> determining if make_recovery was run with the
-C (for check_recovery) option."

if [ -f /var/opt/ignite/recovery/makrec.last 1; then
echo "make-recovery with -C has been run. We'll now run check_recovery."
 /opt/ignite/bin/check_recovery 2>&1
else
echo "make-recovery with -C has not been run. We can't run check_recovery."
fi

} |  tee -a /tmp/IMPORTANT/igtest.out
```

쉘프로그람은 본문파일로서 사용자가 해당한 허락을 가지고 있으면 그 프로그람을 읽거나 수정할수 있다. 쉘프로그람은 #로 시작된 설명행에 프로그람작성정보를 포함할수 있다.

련결

련결(Link)은 체계의 어떤 위치에 기억되여 있는 파일에 대한 지적자이다. 하나의 파일에 대하여 두개이상의 복사를 하지 않고 체계에 이미 존재하는 파일에 련결을 설정할수 있다.

UNIX 에서 련결이 특수하게 리용되는 한가지 방법은 조작체계의 새로운 판에 련관시키는것이다. 파일들의 위치는 때때로 서로 다른 판에서 달라 지며 사용자는 낡은 위치로부터 새 위치에 붙여 진 련결을 리용할수 있다. 그러면 낡은 위치를 리용하는 지령을 수행시킬 때 련결은 새 위치를 가리킨다.

련결은 파일을 집중적으로 관리할 때에도 유용하다. 여러개의 동일한 파일들을 자주 갱신할 때에는 여러 위치에 있는 파일의 사본들을 갱신하는것보다 하나의 중심파일에 련결시키고 그것을 갱신하는것이 더 쉽다.

장치파일

장치파일(Device File)은 때때로 장치규정파일이라고도 부르는데 이 파일은 체계에 접속된 장치에 대한 정보를 포함하는 파일이다. 장치규정파일들은 체계관리기능과 관련되므로 이것은 보통 사용자가 관리하지 않는다. 그러므로 사용자가 플로피디스크나 테프에 파일을 쓰려는 경우에 장치파일에 접근할 방도가 없다. 이 책에서는 몇가지 장치파일을 리용하는 방법을 서술함으로써 이 문제를 해결해 보기로 한다.

체계의 장치들은 보통 여러개의 장치파일로서 접근할수 있다. 실례로 디스크는 블로크장치파일이나 기호장치파일로 접근할수 있다. 이런 접근은 체계관리자의 임무이다. 그러나 사용자도 파일종류를 결정해야 하는 경우에 기호장치나 블로크장치와 같은 각이한 종류의 특수파일들과 만나게 된다.

그밖에도 다른 형태의 파일종류들이 있지만 UNIX 를 처음 다룬다는 점에서는 여기서 서술하는 파일종류만으로도 충분할것이다.

file 지령

file 지령은 파일종류를 결정하는데 쓰인다. 파일의 이름만으로는 그 파일의 종류가 항상 지정되는것이 아니기때문에 file 지령을 쓰는것이 좋다. 다음의 실례에서는 어떤 파일에 대한 배경정보를 렬거해 주며 그다음 파일종류를 보여 주기 위하여 file 지령을 수행시키고 있다. 이 실례에서 파일을 렬거하는데 리용하는 지령에 대해서는 다음항목에서 고찰한다. ls 지령은 파일을 렬거하는 지령이다. 이 장의 뒤에서는 ls 지령에 대한 안내페지를 제시하고 있다. 파일에 대한 렬거는 선택항목 "-l"을 붙여 표시할수도 있다. ls -l 로 되여 있으면 매개 파일의 이름, 파일종류, 허락, 굳은 련결의 개수, 소유자이름, 그룹이름, 바이트수, 시간 등을 렬거한다. 이러한 정보는 file 지령의 출력에서 효과적으로 리용되는 것들이다. 다음의 실례에서는 HP-UX 출력과 Linux 출력을 제시해 준다. 이 실례를 통하여 각이한 UNIX 변종들에서 file 지령의 출력이 어떻게 차이나는가를 알수 있다.

본문파일 (UNIX 실례)

(지령 file에 의하여 ascii본문으로 표시된다.)

```
# ls -1 mosaic-global-history
-rw-r--r--   1 201   users    587  Dec  22      1999  mosaic-global-history
# file  .mosaic-global-history
.mosaic-global-history  ascii  text
#
```

본문파일 (Linux 실례)

(지령 file에 의하여 ASCII본문으로 표시된다.)

```
# ls -l *
-rw-r--r--   1  root   root     251367  Nov    5  07:11  debug
-rw-r--r--   1  root   root     2020    Nov    5  07:11  history
# file  *
debug:      ASCII  text
history:    ASCII  text
```

자료파일 (UNIX 실례)

(지령 file에 의하여 자료로 표시된다.)

```
# ls -1 Static.dat
-rw-r--r-- 1  201     users    235874  Aug  26  1999  Static.dat
# file  Static.dat
Static.dat:  data
#
```

원천코드파일 (UNIX 실례)

(지령 file에 의하여 c프로그람본문으로 표시된다.)

```
# ls -1 krsort.c
-rwxrwxrwx          1  201   users    3234  Nov  16  1999  krsort.c
# file  krsort.c
krsort.c:    c  program  text
#
```

원천코드파일 (Linux 실례)

(지령 file에 의하여 C프로그람본문으로 표시된다.)

```
#  ls -l *.c
-rw-r--r--  1  root  root  4521   Jul  12  1999  intl-bindtextdom.c
-rw-r--r--  1  root  root  6234   Jul  12  1999  intl-cat-compat.c
-rw-r--r--  1  root  root  14128  Jul  12  1999  intl-dcgettext.c
-rw-r--r--  1  root  root  1750   Jul  12  1999  intl-dgettext.c
-rw-r--r--  1  root  root  12759  Jul  12  1999  intl-finddomain.c
-rw-r--r--  1  root  root  1907   Jul  12  1999  intl-gettext.c
-rw-r--r--  1  root  root  1646   Jul  12  1999  intl-intl-compat.c
-rw-r--r--  1  root  root  5361   Jul  12  1999  intl-loadmsgcat.c
-rw-r--r--  1  root  root  7271   Jul  12  1999  intl-localealias.c
-rw-r--r--  1  root  root  2914   Jul  12  1999  intl-textdomain.c
# file *.c
intl-bindtextdom.c: C program text
intl-cat-compat.c:  C program text
intl-dcgettext.c:   C program text
intl-dgettext.c:    C program text
intl-finddomain.c:  C program text
intl-gettext.c:     C program text
intl-intl-compat.c: C program text
intl-loadmsgcat.c:  C program text
intl-localealias.c: C program text
intl-textdomain.c:  C program text
#
```

실행파일 (UNIX 실례)

(지령 file에 의하여 공유된 실행파일로 표시된다.)

```
# ls -1 krsort
-rwxr-xr-x  1  201  users  34592  Nov  16  1999  krsort
# file krsort
krsort:    PA-RISC1.1 shared executable dynamically linked -not stripped
#
```

실행파일 (Linux 실례)

(지령 file에 의하여 실행파일로 표시된다.)

```
# ls -1 a*
-rwxr-xr-x  1  root  root  3888  Jul  24  1999  activate
-rwxr-xr-x  1  root  root  4452  Feb  25  1999  adjtimex
```

```
# file a*
activate:   ELF 32-bit LSB executable, Intel 80386, version 1,   stripped
adjtimex: ELF 32-bit LSB executable, Intel 80386, version 1,   stripped
#
```

쉘프로그람（UNIX 실례）

（지령 file에 의하여 지령본문으로 표시된다.）

```
# ls -1 llsum
-rwxrwxrwx   1   root      Sys     1267 Feb 23    1999 llsum
# file llsum
llsum:                 commands text
#
```

쉘프로그람（Linux 실례）

（지령 file에 의하여 Bourne 쉘스크립트본문으로 표시된다.）

```
# ls -1 request-route
-rwx ------ 1       root     root     1046 Sep 19 1999 request-route
# file request-route
request-route: Bourne shell script text
#
```

련결（UNIX 실례）

（련결은 지령 file에 의하여 참조되지 않는다. 이것은 동적으로 련결되는 공유된 실행파일이라는것을 보여 준다. 동적련결을 참조해 보면 이것이 련결이라는것을 의미하지 않는다.）

```
# ls -1 /usr/bin/ar
lr-xr-xr-t   1     root      sys     15   Mar   23   1999 ar -> /usr/ccs/bin/ar
# file /usr/bin/ar
/usr/bin/ar:           s800 shared executable dynamically linked
#
```

련결（Linux 실례）

（다음의 련결은 기호적련결이다.）

```
# ls -1 reboot
lrwxrwxrwx         1 root  root     4 Nov 5 01:31 reboot  ->  halt
```

file * | grep link

```
depmod:     symbolic  link to  modprobe
ksyms:      symbolic  link to  insmod
pidof:      symbolic  link to  killall5
reboot:     symbolic  link to  halt
rmmod:      symbolic  link to  insmod
swapoff:    symbolic  link to  swapon
telinit:    symbolic  link to  init
udosctl:    symbolic  link to  /sbin/umssync
umssetup:   symbolic  link to  /sbin/umssync
#
```

블로크장치파일(UNIX 실례)

(지령 file에 의하여 특수블로크로 표시된다.)

ls -l /dev/dsk/c0t1d0

```
brw-r--r--    1 bin    sys   31  0x001000  Apr  17  1999  /dev/dsk/c0t1d0
```

file /dev/dsk/c0t1d0

```
/dev/dsk/c0t1d0:    block  special (31/4096)
#
```

블로크장치파일(Linux 실례)

(지령 file에 의하여 특수블로크로 표시된다.)

ls -l loop*

```
brw-rw ----1   root   disk   7,   0   Sep   23  1999   loop0
brw-rw ----1   root   disk   7,   1   Sep   23  1999   loop1
brw-rw ----1   root   disk   7,   2   Sep   23  1999   loop2
brw-rw ----1   root   disk   7,   3   Sep   23  1999   loop3
brw-rw ----1   root   disk   7,   4   Sep   23  1999   loop4

brw-rw ----1   root   disk   7,   5   Sep   23  1999   loop5
brw-rw ----1   root   disk   7,   6   Sep   23  1999   loop6
brw-rw ----1   root   disk   7,   7   Sep   23  1999   loop7
```

file loop*

```
loop0:  block    special (7/0)
loop1:  block    special (7/1)
loop2:  block    special (7/2)
loop3:  block    special (7/3)
```

loop4: block special（7/4）
loop5: block special（7/5）
loop6: block special（7/6）
loop7: block special（7/7）
#

기호장치파일（UNIX 실례）

（지령 file에 의하여 특수기호로 표시된다.）

 # ls -l /dev/rdsk/c0t1d0
crw-r ----- 1 root sys 188 0x001000 Mar 23 1999 /dev/rdsk/c0t1d0
 # file /dev/rdsk/c0tid0
/dev/rdsk/c0tld0: character special（188/4096）
#

기호장치파일（Linux 실례）

（지령 file에 의하여 특수기호로 표시된다.）

 # ls -l mi*
crw-rw-rw- 1 root sys 14, 2 Sep 23 1999 midi00
crw-rw-rw- 1 root sys 14, 18 Sep 23 1999 midi01
crw-rw-rw- 1 root sys 14, 34 Sep 23 1999 midi02
crw-rw-rw- 1 root sys 14, 50 Sep 23 1999 midi03
crw-rw-rw- 1 root sys 14, 0 Sep 23 1999 mixer
crw-rw-rw- 1 root sys 14, 16 Sep 23 1999 mixer1
 # file mi*
midi00: character special（14/2）
midi01: character special（14/18）
midi02: character special（14/34）
midi03: character special（14/50）
mixer: character special（14/0）
mixer1: character special（14/16）
#

ls 지령

ls 지령에 대해서는 론의할것이 많다. 이미 file 지령을 설명하면서 ls 지령의 선택항목 -l에 대하여 고찰하였지만 ls 지령이 없으면 UNIX체계에서 많은 일을 할수 없다. 이 지령의 중요성을 실례를 통하여 고찰하자.

ls

다음의 실례에서는 등록부외에 아무런 선택항목도 없이 ls 지령을 수행시킨 결과를 보여 준다.

```
# ls /home/denise

27247b.exe
410ptl.exe
410pt2.exe
41ndir.exe
41nds1.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail
N3212B6.EXE
SCSI4S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en0316bz.exe
en0316tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpd10117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
ja95up.exe
msie10.exe
n32e12n.exe
```

nfs197.exe
pass.sb
plusdemo.exe
ps4x03.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe
roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp41a.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe
total 46718

우의 실례에서 어느것이 파일이고 어느것이 등록부인가? 모든 구체레가 다 렬거되였는가? ls에는 이런 물음에 대답을 주는 여러개의 선택항목이 있다.

ls 지령은 규정된 등록부의 내용을 렬거한다. 그리고 등록부가 규정되지 않으면 현재 작업등록부의 내용을 현시한다.

ls -a

어떤 등록부의 모든 구체레를 현시하려면 선택항목 -a 를 리용한다. "."으로 시작된 파일들은 숨은 파일들이며 보통 ls에 의하여 현시되지 않는다. 다음의 실례는 ls -a의 출력을 보여 준다.

$ **ls -a /home/denise**
.Xauthority
.cshrc
.dt
.dtprofile
.elm

.exrc
.fmrc.orig
.glancerc
.qpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-cookies
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
.profile
.rhosts
.sh_history
.softbuildrc
.softinit.orig
.sw
.xinitrc
.xsession
27247b.exe
410ptl.exe
410pt2.exe
41ndir.exe
41ndsl.exe
41nds4.exe
41nwad.exe
41rtr2.exe
HPDA1.EXE
Mail
N3212B6.EXE
SCS14S.EXE
clean
clean2

clean3
content.exe
dsenh.exe
egl
eg2
en03l6bz.exe
en03l6tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpdl0117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
ja95up.exe
msie10.exe
n32el2n.exe
nfsl97.exe
pass.sb
plusdemo.exe
ps4xO3.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe
roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp4la.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe
total 46718

우의 실레에서 보는것처럼 숨은 파일들은 "."으로 시작되였으며 모두 현시되였다. 숨은 파일들은 -a선택항목이 없으면 현시되지 않는다.

ls -l

등록부안의 전체 정보를 현시하려면 선택항목 -l 을 사용한다(아래의 실례에서 일부 파일이름들은 페지면상의 제한으로 잘리웠다.).

$ ls -al /home/denise

```
-rw-------    1  denise  users     98  Oct   6  09:19  .Xauthority
-r--r--r--    1  denise  users    814  May  19  10:10  .cshrc
drwxr-xr-x    7  denise  users   1024  Sep  26  11:14  .dt
-rwxr-xr-x    1  denise  users   8705  Jul   7  12:04  .Atprofile
drwx------    2  denise  users   1024  Jul  31  18:48  .elm
-r--r--r--    1  denise  users    347  May  19  10:10  .exrc
-rwxrwxrwx    1  denise  users    170  Jun   6  14:20  .fmrc.orig
-rw-------    1  denise  users     97  Jun  12  18:59  .glancerc
-rw-------    1  denise  users  17620  Sep  21  16:11  .gpmhp
-rwxr-xr-x    1  denise  users    391  Sep  19  09:55  .history
-r--r--r--    1  denise  users    341  May  19  10:10  .login
drwx--x—x     2  denise  users   1024  Jul  31  18:48  .1rom
-rw-r--r--    1  denise  users    768  Jul  28  12:54  .mailrc
-rw-------    1  denise  users   1450  Oct   6  13:58  .netscapeboo-
drwx------    2  denise  users  10240  Oct  10  15:24  .netscapecache
-rw-------    1  denise  users     91  Sep  18  14:16  .netscapecoo-
-rw-------    1  denise  users  43906  Oct  10  15:32  .netscapehi-
-rw-r--r--    1  denise  users    566  Aug  25  14:3 6  .netscapene-
-rw-------    1  denise  users  46514  Jun  28  12:35  .netscape.hp-
-rw-------    1  denise  users   1556  Sep  28  15:02  .netscapepre-
-rw-------    1  denise  users    104  Jul  11  11:01  .newsrc-news
-rw-r--r--    1  denise  users    223  Sep  26  13:26  .newsrcnewv
-r--r--r--    1  denise  users    446  May  19  10:10  .profile
-rw-------    1  denise  users     21  Jul   6  13:21  .rhosts
-rw-------    1  denise  users   2328  Oct  10  15:22  .sh_history
-rw-r--r--    1  denise  users   1052  Sep  22  15:00  .softbuildrc
-rwxrwxrwx    1  denise  users    161  Jul  11  12:19  .softinit.orig
drwxr-xr-x    3  denise  users   1024  Aug  31  15:44  .sw
-rw-------    1  denise  users     23  Jun   2  15:01  .xinitrc
```

```
-rwxr-xr-x   1 denise users   11251 May 19 10:41 .xsession
-rw-r--r--   1 denise users  611488 Oct  3 12:00 27247b.exe
-rw-r--r--   1 denise users  114119 Sep 29 12:49 410ptl.exe
-rw-r--r--   1 denise users  136979 Sep 29 12:53 410pt2.exe
-rw-r--r--   1 denise users  173978 Sep 29 12:40 41ndir.exe
-rw-r--r--   1 denise users  363315 Sep 29 12:52 41ndsi.exe
-rw-r--r--   1 denise users  527524 Sep 29 12:57 41nds4.exe
-rw-r--r--   1 denise users 1552513 Sep 29 12:50 41nwad.exe
-rw-r--r--   1 denise users  853424 Sep 29 12:24 41rtr2.exe
-rw-r--r--   1 denise users 1363011 Sep 20 12:20 HPDA1.EXE
drwx------   2 denise users      24 Jul 31 18:48 Mail
-rw-r--r--   1 denise users 1787840 Aug 31 09:35 N3212B6.EXE
-rw-r--r--   1 denise users   13543 Sep 23 09:46 SCS14S.EXE
-rw-r--r--   1 denise users   28395 Aug 30 15:07 cabview.exe
-rwx--x--x   1 denise users      66 Jun  8 17:40 clean
-rwx--x--x   1 denise users      99 Jun 20 17:44 clean2
-rwx--x--x   1 denise users      66 Jun 20 17:51 clean3
-rw-r--r--   1 denise users   15365 Aug 30 15:07 content.exe
-rw-r--r--   1 denise users  713313 Sep 29 12:56 dsenh.exe
-rwx------   1 denise users     144 Aug 14 17:10 egl
-rwx------   1 denise users     192 Aug 15 12:13 eg2
-rw-r--r--   1 denise users  667890 Sep 20 12:41 en0316bz.exe
-rw-r--r--   1 denise users  641923 Sep 20 12:42 en03l6tb.exe
-rw-r--r--   1 denise users    6251 Aug 30 15:07 explore.exe
-rw-r--r--   1 denise users   23542 Aug 30 15:08 flexi_cd.exe
-rw-r--r--   1 denise users      30 Aug 14 17:02 fred.h
-rw-r--r--   1 denise users       0 Aug 14 17:24 hal.c
-rw-r--r--   1 denise users  895399 Sep 20 12:32 hpdlOI17.exe
-rw-r--r--   1 denise users   14135 Sep 20 12:39 hpdlinst.txt
-rw-------   1 denise users    2943 Jun 19 14:42 hpux.patches
-rw-r--r--   1 denise users  680279 Sep 20 12:26 j2577a.exe
-rw-r--r--   1 denise users  930728 Sep 20 15:16 ja95up.exe
-rw-r--r--   1 denise users   53575 Oct 10 10:37 mbox
-rw-r--r--   1 denise users 1097728 Aug 30 15:03 MBielO.exe
-rw-r--r--   1 denise users 1790376 Sep 18 14:32 n32el2n.exe
-rw-r--r--   1 denise users 1393835 Sep 29 12:59 nfsl97.exe
-rw-------   1 denise users     977 Jul  3 14:25 pass.sb
-rw-r--r--   1 denise users 1004544 Aug 30 15:00 plusdemo.exe
-rw-r--r--   1 denise users  229547 Sep 29 12:27 ps4xO3.exe
```

```
-rwxr--r--   1  denise  users      171  Aug   9  13:43  psg
-rw-r--r--   1  denise  users    16645  Aug  30  15:08  quik es.exe
-rw-r--r--   1  denise  users    14544  Aug  30  15:08  rclo-jk.exe
-rw-r--r--   1  denise  users  2287498  Aug  30  15:12  rkhelp.exe
-rw-r--r--   1  denise  users        0  Aug  15  12:10  roni.mak
-rw-r--r--   1  denise  users     1139  Sep  28  10:35  sb.txt
-rw-r--r--   1  denise  users   569855  Sep  29  12:55  smsup2.exe
-rw-------   1    root    Sys      161  Jul  11  12:18  softinit.remoteso
-rw-r--r--   1  denise  users       39  Sep  29  12:48  srvpr.exe
-rw-r--r--   1  denise  users       38  Aug  15  12:14  steve.h
-rw-r--r--   1  denise  users    14675  Aug  30  15:08  target.exe
-rw-r--r--   1  denise  users   229630  Sep  29  12:54  tcp4la.exe
-rw-r--r--   1  denise  users  1954453  Sep  29  12:26  tnds2.exe
-rw-r--r--   1  denise  users   364270  Sep  23  09:50  upgrade.exe
-rwx-----x   1  denise  users       88  Aug   9  13:43  whoon
-rw-r--r--   1  denise  users   191495  Aug  30  15:00  win95app.exe
total  46718
```

ls -l 을 사용하면 모든 파일의 매개 마당정보가 다 나온다. 그러나 파일이름을 규정하면 그 파일에 대한 정보만 나온다.

```
$ ls -l sort
-rwxr-x--x   1   marty  users  120  Jul  26  10:20  sort
```

첫번째 마당은 그 파일에 대한 접근허락정보를 정의한다. 이것은 이미 앞에서 허락으로 언급한것이다. 이 접근허락정보를 보면 소유자는 그 파일에 대한 읽기, 쓰기, 실행허락을 가진다. 그러나 그룹성원들은 읽기 및 실행허락만을 가지며 다른 사람들은 실행허락만을 가진다.

두번째 마당은 련결개수이다. 이것은 몇개의 파일이 그 파일에 기호적으로 련결되여 있는가를 현시한다. 파일을 련결시키는 ln 지령에 대해서는 후에 자세히 보기로 한다. 우의 실례에서 련결개수는 한개 즉 이 파일이 자기자체에만 련결되여 있다는것을 보여 준다. 아래 실례에서 보여 주는 .dt 등록부에 대한 보조등록부의 개수는 련결개수와 약간 다르다. 보조등록부개수에는 주어 진 등록부자체와 그것의 어미등록부까지 포함된다. .dt 아래에 있는 5개의 보조등록부들은 다음과 같다.

```
drwxr-x--x   7   denise  users  1024  Jul  26  10:20  .dt
```

/home/denise/.dt 아래에 있는 보조등록부들은 다음과 같다.

/home/denise/A/Desktop
/home/denise/At/appmanager
/home/denise/. dt/palettes
/home/denise/. dt/sessions
/homeldenise/. dt/types

우와 같은 5 개의 보조등록부와 주어 진 등록부자체 그리고 그것의 어미등록부를 모두 합하여 7 개이다.

세번째 마당은 파일소유자를 현시한다. 여기서는 가입이름(실례에서는 denise)이 현시된다. 파일을 창조할 때 가입이름이 기정소유자로 규정된다.

네번째 마당은 그 파일이 속하는 그룹의 이름이다. 그룹에 대해서는 이미 "허락"에서 서술하였다.

다섯번째 마당은 파일의 크기를 표시한다. Sort 파일의 크기는 120byte 이다.

여섯번째 마당은 파일이 창조되거나 마지막으로 변경된 날자와 시간을 표시한다.

일곱번째 마당은 파일과 등록부를 자모순으로 현시한다. 먼저 "."으로 시작된 파일들이 현시되고 그 다음에는 수자로 시작된 파일들이 현시되며 그 뒤에 대문자로 시작된 파일들, 최종적으로는 소문자로 시작된 파일들이 현시된다.

UNIX 에서 파일이름의 첫 기호로 시작되는 기호가 여러개 있을수 있으므로 ls -l 로 파일이름을 렬거할 때 자기가 찾는 파일을 찾기 힘들수 있다.

ls -i

파일에 대한 색인마디정보를 얻으려면 ls 지령과 함께 선택항목 -i 를 사용한다. 다음 실례는 선택항목 -i 및 -l 을 사용한것이다.

$ ls -ail /home/denies

```
137717  -rw-------  1  denise  users     98  Oct   6  09:19  . Xauthority
137623  -r--r--r--  1  denise  users    814  May  19  10:10  . cshrc
140820 drwxr-xr-x   7  denise  users   1024  Sep  26  11:14  . dt
137629 -rwxr-xr-x   1  denise  users   8705  Jul   7  12:04  . dtprofile
180815 drwx------   2  denise  users   1024  Jul  31  18:48  . elm
137624 -r--r--r--   1  denise  users    347  May  19  10:10  . exrc
137652 -rwxrwxrwx1     denise  users    170  Jun   6  14:20  . fmrc.orig
137650 -rw-------   1  denise  users     97  Jun  12  18:59  . glancerc
137699 -rw-------   1  denise  users  17620  Sep  21  16:11  . gpmhp
137640 -rwxr-xr-x   1  denise  users    391  Sep  19  09:55  . history
137625 -r--r--r--   1  denise  users    341  May  19  10:10  . login
185607 drwx--x--x   2  denise  users   1024  Jul  31  18:48  . 1rom
```

```
137642 -rw-r--r--   1  denise  users      768  Jul  28  12:54  . mailrc
137641 -rw-------   1  denise  users     1450  Oct   6  13:58  . netscaperks.html
179207 drwx ------  2  denise  users    10240  Oct  10  15:24  . netscape-cache
137656 -rw-------   1  denise  users       91  Sep  18  14:16  . netscape-cookies
137635 -rw-------   1  denise  users    43906  Oct  10  15:32  . netscape-history
137645 -rw-r--r--   1  denise  users      566  Aug  25  14:36  . netscapes.spry.com
137646 -rw-------   1  denise  users    46514  Jun  28  12:35  . netsca
137634 -rw-------   1  denise  users     1556  Sep  28  15:02  . netscaperences
137637 -rw-------   1  denise  users      104  Jul  11  11:01  . newsrcws.spry.c!om
137633 -rw-r--r--   1  denise  users      223  Sep  26  13:26  . newsrc-hp.com
137626 -r--r--r--   1  denise  users      446  May  19  10:10  . profile
137649 -rw-------   1  denise  users       21  Jul   6  13:21  . rhosts
137694 -rw-------   1  denise  users     2328  Oct  10  15:22  . sh-history
137698 -rw-r--r--   1  denise  users     1052  Sep  22  15:00  . softbuildrc
137636 -rwxrwxrwx1     denise  users      161  Jul  11  12:19  . softinit.orig
33600 drwxr-xr-x    3  denise  users     1024  Aug  31  15:44  . sw
137648 -rw-------   1  denise  users       23  Jun   2  15:01  . xinitrC
137628 -rwxr-xr-x   1  denise  users    11251  May  19  10:41  . xsession
137715 -rw-r--r--   1  denise  users   611488  Oct   3  12:00  27247b.exe
137707 -rw-r--r--   1  denise  users   114119  Sep  29  12:49  410ptl.exe
137710 -rw-r--r--   1  denise  users   136979  Sep  29  12:53  410pt2.exe
137705 -rw-r--r--   1  denise  users   173978  Sep  29  12:40  41ndir.exe
137709 -rw-r--r--   1  denise  users   363315  Sep  29  12:52  41ndsi.exe
137714 -rw-r--r--   1  denise  users   527524  Sep  29  12:57  41nds4.exe
137708 -rw-r--r--   1  denise  users  1552513  Sep  29  12:50  41nwad.exe
137696 -r -r--r--   1  denise  users   853424  Sep  29  12:24  41rtr2.exe
137654 -rw-r--r--   1  denise  users  1363011  Sep  20  12:20  HPDAI.EXE
182429 drwx------   2  denise  users       24  Jul  31  18:46  Mail
137683 -rw-r--r--   1  denise  users  1787840  Aug  31  09:35  N3212B6.EXE
137702 -rw-r--r--   1  denise  users    13543  Sep  23  09:46  SCS14S.EXE
137638 -rwx--x--x   1  denise  users       66  Jun   8  17:40  clean
137651 -rwx--x--x   1  denise  users       99  Jun  20  17:44  clean2
137632 -rwx--x--x   1  denise  users       66  Jun  20  17:51  clean3
137688 -rw-r--r--   1  denise  users    15365  Aug  30  15:07  content.exe
137713 -rw-r--r--   1  denise  users   713313  Sep  29  12:56  dBenh.exe
137667 -rwx------   1  denise  users      144  Aug  14  17:10  eg1
137671 -rwx------   1  denise  users      192  Aug  15  12:13  eg2
137662 -rw-r--r--   1  denise  users   667890  Sep  20  12:41  en03l6bz.exe
137665 -rw-r--r--   1  denise  users   641923  Sep  20  12:42  en0316tb.exe
```

```
137689 -rw-r--r--  1  denise  users     6251  Aug  30  15:07  explore.exe
137690 -rw-r--r--  1  denise  users    23542  Aug  30  15:08  flexi_cd.exe
137670 -rw-r--r--  1  denise  users       30  Aug  14  17:02  fred.h
137673 -rw-r--r--  1  denise  users        0  Aug  14  17:24  hal.c
137660 -rw-r--r--  1  denise  users   895399  Sep  20  12:32  hpdlOll7.exe
137661 -rw-r--r--  1  denise  users    14135  Sep  20  12:39  hpdlinst.txt
137647 -rw-------  1  denise  users     2943  Jun  19  14:42  hpux.patches
137659 -rw-r--r--  1  denise  users   680279  Sep  20  12:26  j2577a.exe
137697 -rw-r--r--  1  denise  users   930728  Sep  20  15:16  ja95up.exe
137684 -rw-r--r--  1  denise  users  1097728  Aug  30  15:03  msie10.exe
137658 -rw-r--r--  1  denise  users  1790376  Sep  18  14:32  n32el2n.exe
137643 -rw-r--r--  1  denise  users  1393835  Sep  29  12:59  nfs197.exe
137639 -rw-------  1  denise  users      977  Jul   3  14:25  pass.sb
137664 -rw-r--r--  1  denise  users  1004544  Aug  30  15:00  plusdemo.exe
137704 -rw-r--r--  1  denise  users   229547  Sep  29  12:27  ps4xO3.exe
137666 -rwxr--r--  1  denise  users      171  Aug   9  13:43  psg
137691 -rw-r--r--  1  denise  users    16645  Aug  30  15:08  quik_res.exe
137692 -rw-r--r--  1  denise  users    14544  Aug  30  15:08  rclock.exe
137693 -rw-r--r--  1  denise  users  2287498  Aug  30  15:12  rkhelp.exe
137669 -rw-r--r--  1  denise  users        0  Aug  15  12:10  roni.mak
137657 -rw-r--r--  1  denise  users     1139  Sep  28  10:35  sb.txt
137712 -rw-r--r--  1  denise  users   569855  Sep  29  12:55  smsup2.exe
137644 -rw-------  1    root    sys      161  Jul  11  12:18  softinittesoftcm
137706 -rw-r--r--  1  denise  users       39  Sep  29  12:48  srvpr.exe
137682 -rw-r--r--  1  denise  users       38  Aug  15  12:14  steve.h
137685 -rw-r--r--  1  denise  users    14675  Aug  30  15:08  target.exe
137711 -rw-r--r--  1  denise  users   229630  Sep  29  12:54  tcp4la.exe
137700 -rw-r--r--  1  denise  users  1954453  Sep  29  12:26  tnds2.exe
137703 -rw-r--r--  1  denise  users   364270  Sep  23  09:50  upgrade.exe
137663 -rwx-----x  1  denise  users       88  Aug   9  13:43  whoon
137678 -rw-r--r--  1  denise  users   191495  Aug  30  15:00  win95app.exe
```

색인마디번호에는 디스크에서 파일 및 등록부의 위치, 접근허락, 소유자 및 그룹 ID, 파일련결수, 최종수정시간, 최종접근시간, 특수파일에 대한 장치식별번호, 기타 여러가지 정보가 포함된다. 색인마디번호는 사용자가 등록부를 변경시키거나 여러가지 과제를 수행할 때 체계가 전면적으로 리용하는 번호이다.

ls -p

사용자가 지적하는 등록부안에는 많은 보조등록부들이 있을수 있기때문에 등록부를 현시할 때 보조등록부들의 이름뒤에는 사선 "/"을 붙여 편리하게 구분해 보도록 하고 있다. 이 기능은 선택항목 -p로서 수행된다.

$ **ls -ap /home/denise**

```
.Xauthority
.C  shrc
.dt/
.dtprofile
.elm/
.exrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom/
.mailrc
.netscape-bookmarks.html
.netscape-cache/
.netscape-cookies
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
.profile
.rhosts
.sh_history
.softbuildrc
.softinit.orig
.sw/
.xinitrc
.xsession
27247b.exe
```

410ptl.exe
410pt2.exe
41ndir.exe
41ndsl.exe
41nds4.exe
41nwad.exe
41rt r2.exe
HPDAI.EXE
Mail/
N3212B6.EXE
SCS14S.EXE
clean
clean2
clean3
content.exe
dsenh.exe
eg1
eg2
en03l6bz.exe
en03l6tb.exe
explore.exe
flexi_cd.exe
fred.h
hal.c
hpd10117.exe
hpdlinst.txt
hpux.patches
j2577a.exe
ja95up.exe
msie10.exe
n32e12n.exe
nfs197.exe
pass.sb
plusdemo.exe
ps4x03.exe
psg
quik_res.exe
rclock.exe
rkhelp.exe

roni.mak
sb.txt
smsup2.exe
softinit.remotesoftcm
srvpr.exe
steve.h
target.exe
tcp4la.exe
tnds2.exe
upgrade.exe
whoon
win95app.exe

ls -R

사용자가 현시하려는 보조등록부들은 그 아래에 파일 및 보조등록부들을 더 가질수 있기때문에 이것들은 재귀적으로 현시하도록 할수 있다. 선택항목 -R 는 바로 이러한 재귀적인 현시를 보장한다. 아래의 실례에서 준 /home/denise 아래의 모든 보조등록부는 너무 길어서 좀 잘리웠다.

$ **ls -&R /home/denize**

.Xauthority
.cshrc
.dt
.dtprofile
.elm
.exrc
.fmrc. orig
.glancerc
.gpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-cookies
.netscape-history

. netscape-newsgroups-news. spry. com
. netscape-newsgroups-newsserv. hp. com
. netscape-preferences
. newsrc-news. spry. com
. newsrc-newsserv. hp. com
. profile
. rhosts
. sh_history
. softbuildrc
. softinit. orig
. sw
. xinitrc
. xsession
27247b. exe
410ptl. exe
410pt2. exe
41ndir. exe
41ndsl. exe
41nds4. exe
41nwad. exe
41rtr2. exe
HPDAI. EXE
Mail
N3212B6. EXE
SCS14S. EXE
clean
clean2
clean3
content. exe
dsenh. exe
eg1
eg2
en03l6bz. exe
en03l6tb. exe
explore. exe
flexi_cd. exe
fred. h

(현시내용을 일부 생략하였다.)

/home/denise/. lrom:

LRAAAa27637.CC
LRBAAa27637.CC
LROM.AB
LROM.AB.OLD
LROM.AB.lk1
LROM.SET

/home/denise/.netscape-cache:
cache306C05510015292.gif
cache306C05560025292.gif
cache306C05560035292.gif

/home/denise/.sw:
sessions

(나머지 현시내용을 생략하였다.)

ls 개요

지금까지는 ls 에서 제일 중요하고 자주 리용되는 선택항목들만 보았다. 그러나 ls 지령에 대한 소개를 통하여 보다 많은 정보를 얻을수 있다.

아래에 많이 쓰이는 ls 선택항목들에 대한 개요를 준다.

- -a 모든 구체례들을 다 현시한다.
- -b 도형이 아닌 기호들을 인쇄한다.
- -c 파일을 마지막으로 수정된 시간순서대로 현시된다.
- -d 등록부이름만 현시한다. 다른 내용은 현시되지 않는다.
- -f 매개 인수들을 등록부라고 가정한다.
- -g 소유자는 아니고 그룹만이 인쇄된다.
- -i 첫 렬에 색인마디번호를 인쇄한다.
- -m 반점으로 구분하면서 화면의 가로방향으로 인쇄한다.
- -n 이름대신에 UID 및 GID 번호가 인쇄된다.
- -o 그룹이 생략된다는것을 제외하고는 긴 형식(-l)으로 정보를 현시하는것과 동일하다.
- -p 등록부이름뒤에 사선(/)을 붙인다.

-q　　인쇄할수 없는 기호는 "?"로 표시한다.

-r　　파일이 인쇄되는 순서가 반대로 되게 한다.

-s　　바이트가 아니라 블로크단위로 크기를 표시한다.

-t　　최근시간을 먼저 놓으면서 보관된 순서로 표시한다.

-u　　최종수정시간이 아니라 최종접근시간의 순서대로 파일이 인쇄된다.

-x　　실례처럼 다중렬형식으로 파일들을 현시한다.

-A　　현재등록부 및 어미등록부가 현시되지 않는다는것을 제외하고는 -a와 같다.

-C　　다중렬출력형식으로 현시한다.

-F　　등록부는 "/", 실행파일은 "*", 기호적련결은 "@", FIFO 는 "|"을 뒤에 붙인다.

-L　　련결이 가리키는 파일이나 등록부를 현시한다.

-R　　재귀적으로 보조등록부들을 현시한다.

-l　　단일렬형식으로 출력이 진행된다.

짧게 속기형식으로 지령들을 쓸수도 있다. 실례로 ll 은 ls -l 과 동일하며 lsr 는 ls -R와 동일하다. 별명을 창조하여 자기 식의 속기지령들을 만들수도 있다. 이에 대해서는 뒤장에서 고찰하기로 한다.

파일과 관련된 지령들은 또한 **통용기호**(Wildcard)로도 쓸수 있다. 다음항목에서는 파일이름의 확장과 통용기호에 대하여 고찰한다.

파일체계구조

반드시 하나의 **파일체계**(File System)만을 가지고 작업해야 한다는 원칙은 없다. 체계관리자는 자기 체계에 여러가지 파일체계를 설치할수 있다. 초학자라면 파일체계의 여러가지 형태에 대해 지나치게 관심할 필요는 없지만 파일체계구조에 들어가기에 앞서 간단히 몇개의 파일체계를 고찰하는것이 좋다.

사용자가 입력하는 일부 지령들은 파일체계종류에 따라 "-F"와 같은 선택항목을 규정할것을 요구하기때문에 체계관리자는 여러가지 형태의 파일체계를 관리하여야 한다. UNIX 변종들이 어떤 파일체계들을 제공하는가를 알아 내는 제일 좋은 방법은 mount 지령에 대한 지령소개를 리용하는것이다. 선택항목 "-F"나 Linux 의 선택항목 "-t"는 제공되는 파일체계종류들의 목록뒤에 붙는다. UNIX 체계의 변종에 따라 해당한 파일체계와 그 파일체계에 따르는 지령선택항목들은 서로 다르다. 보통 파일체계종류선택항목을 제공하는 지령들은 dcopy, fsck, mksf, mount, newfs 등이다. 앞으로 체계관리자로서의 역할을 하자면 이러한 지령들을 알아야 할 필요가 있다. 아래에 UNIX 변종들이 공통적으로 지원하는 파일체계들을 보여 준다.

- **PC 파일체계**(PCFS: Personal Computer File System)는 PC 형식화디스크들에 직접 접근할수 있게 하는 파일체계이다.
- **UNIX 파일체계**(UFS: UNIX File System)는 여러가지 UNIX 변종에서 사용하는 표준적이며 기본적인 파일체계이다.
- **CD-ROM 파일체계**(CDFS: CD-ROM File System)는 CD-ROM 을 설치할 때 리용한다. 대부분의 CD-ROM 은 읽기만 가능하고 쓰기는 불가능하다. 일부 UNIX 체계들에서는 이 파일체계를 높은 산줄기파일체계(HSFS: High Sierra File System)라고 부르기도 한다.
- **망파일체계**(NFS: Network File System)는 국부체계로부터 망에 있는 다른 체계의 파일을 접근하도록 한다. NFS 가 설치된 파일체계는 그 파일체계가 다른 체계에 위치하고 있다고 하여도 자기의 체계에서 국부적체계처럼 보인다.
- **되돌림파일체계**(LOFS: Loopback File System)는 동일한 파일체계를 서로 엇갈리는 경로이름을 리용하여 여러 장소에서 사용하도록 한다.
- VxFS 는 파일체계에 대한 빠른 복구와 HP-UX 우에서의 여벌복사와 같은 직결기능들을 보장하는 **확장구역기초 실행기록파일체계**(extent-based Journal File System)이다.
- **고성능파일체계**(HFS: High Performance File System)는 UNIX 파일체계의 HP 방안이다. 이 고성능파일체계는 대부분의 실례들에서 리용된다.
- TMPFS 는 기억기에 기초한 파일체계(**기억기토대파일체계**(Memory-Based File System))이다.
- CacheFS 는 고속완충기에 존재하는 파일체계(**고속완충기파일체계**(Cache File System))이다.

파일체계의 종류는 UNIX 변종들에서 강하게 전용화되는 대상이다. UNIX 와 그것의 관련지령들이 대단히 류사하다고 해도 파일체계는 일반적으로 그 UNIX 변종에 고유한것이다. 그러므로 mount 지령의 안내페지를 보고 해당한 UNIX 체계의 파일체계형태가 무엇인가를 결정해야 한다.

이 책에서는 Caldera Linux 체계의 mount 안내페지의 파일체계종류들을 많이 리용하고 있다. Linux 의 mount 지령에서 -t vfstype 선택항목을 리용하여 설치하려는 하나의 파일체계를 규정할수 있다.

- minix
- ext
- ext2
- Xiafs
- hpfs
- fat
- msdos
- umsdos
- vfat

- proc
- nfs
- iso9660
- smb
- ncp
- affs
- Ufs
- sysv
- xenix
- coherent

Linux 체계에서 자주 설치하는 공통적인 파일체계는 DOS 플로피디스크체계이다. 아래의 실례에서는 DOS 플로피체계의 설치와 이 체계에 파일의 복사, 이 체계의 해제과정을 보여 주고 있다.

mount -t msdos /dev/fd0 /mnt/floppy
cp * /mnt/floppy
 # ls /=t/floppy
file1 file2 file3 file4
umount /dev/fd0

우의 실례에서는 먼저 /dev/fd0 을 설치한다. 이 /dev/fd0 은 플로피디스크장치로서 /mnt/floppy 아래에 msdos 형으로 설치된다. 왜냐하면 -t msdos 로 규정되였기때문이다. 다음에는 현등록부의 모든 파일을 플로피디스크에 복사한다. 다음에는 플로피디스크에 있는 모든 파일들(4 개)이 ls 에 의하여 현시된다. 그리고 unmount 지령으로 플로피디스크를 해제시킨다. 그러면 플로피디스크를 DOS 체계로 취할수 있고 파일을 읽을수 있다. 플로피디스크가 msdos 형태로 설치되였기때문에 파일은 DOS 형식으로 플로피디스크에 씌여졌다.

류사한 방법으로 CD-ROM 을 Linux 체계에 설치하려면 mount 지령에 의하여 다음과 같이 할수 있다.

mount /dev/hdd /cdrom
mount: block device /dev/hdd is write-protected, mounting readonly
#

CD-ROM 장치파일은 /dev/hold 로 할수 있다. 이 장치는 물론 mount 에서 지적하는것처럼 읽기전용장치이다. UNIX 변종들의 파일체계구조는 대부분 AT&T SVR4 구조에 기초하고 있다. 때문에 UNIX 체계들에서 리용하는 파일체계에서는 동일한 이름들을 많이 사용하고 있다. 그림 2−1 에서는 높은 준위의 파일체계에 대한 표시를 주고 있다.

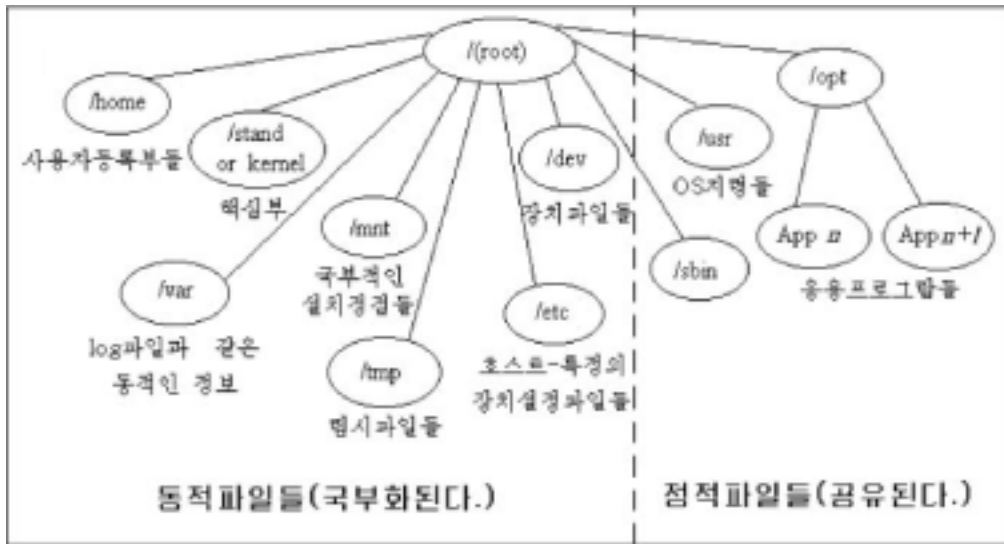



그림 2-1. 일반적인 UNIX 파일체계구조(UNIX 변종들에서는 약간 다르다.)

UNIX 파일체계구조에는 다음과 같은 몇가지 중요한 기능들이 있다.

- 파일 및 등록부는 부류별로 조직된다. 제일 뚜렷한 두 부류가 있는데 그림 2-1에서 보여 주는바와 같이 그 두 부류는 **정적파일**(Static File)들과 **동적파일**(Dynamic File)들이다. 물론 실행파일, 환경파일, 자료파일과 같은 다른 부류도 있을수 있다. 정적파일들을 망에 있는 다른 **주콤퓨터**(Host Computer)들이 공유할수 있기때문에 정적파일들에는 "shared"라는 표식이 붙는다. /usr, /sbin, /opt는 공유등록부들이다.
- 조작체계와 응용프로그람은 서로 분리되여 보관된다. 응용프로그람판매자들은 그 프로그람이 어디에 적재되는가에 관계하지 않는다. 그러나 체계관리자는 응용프로그람을 조작체계와 분리시켜 보관하는데 큰 관심을 돌림으로써 사용자들이 응용프로그람파일이 무의식적으로라도 조작체계파일과 겹치지 않도록 하고 있다. 또한 응용프로그람들이 분리된 구역에 적재되면 그것들이 모듈적이기때문에 편리하다. 모듈적이라는 의미는 체계관리자가 다른 응용프로그람에 영향을 주지 않으면서 응용프로그람을 추가하거나 제거, 수정할수 있다는것이다. 응용프로그람은 /opt 등록부에 보관된다.
- 내부적인 체계파일들은 체계내부나 망접근파일들과 분리되여 보관된다. /usr 및 /sbin 은 공유되는 조작체계등록부이다. 주콤퓨터의 고유한 정보는 이 두개의 등록부에 포함되지 않는다. /etc 등록부는 주콤퓨터의 고유한 환경파일들을 보관하는데 리용된다.
- 실행파일들도 체계의 환경파일과 분리되여 보관되므로 실행파일들은 주콤퓨터들에서 공유될수 있다. 환경파일들을 그것들을 리용하는 프로그람과 분리시킨것은 조작체계에 대한 갱신이 환경파일에 영향을 주지 않는다는것을 의미한다.

아래에 파일체계구조에서 제일 중요한 몇개의 등록부와 그것의 내용에 대하여 서술한다.

/ 이것은 **뿌리등록부**(Root Directory)이다. 뿌리등록부는 파일체계의 **계층나무구조**(Hierarchical Tree Structure)에서 토대로 되는 등록부이다. 등록부는 론리적으로 /의 부분으로서 고찰된다. 디스크에 어떤 등록부나 론리적인 기록권(Volumn)이 기억되여 있는가에는 무관계하게 모든 등록부는 뿌리계층의 한 부분으로 론리적으로 고찰된다.

/dev 이것은 주콤퓨터에 고유한 장치파일들을 포함한다.

/etc 이것은 주콤퓨터에 고유한 체계 및 응용프로그람환경파일들을 포함한다. 이 등록부안의 정보는 체계의 조작에서 매우 중요하며 영원히 본질적인 정보로 된다. /etc 아래에는 보충적인 환경등록부들도 있다. 특히 중요한것은 두개의 /etc 보조등록부들이 있다는것이다.

/home 이것은 사용자의 **홈등록부**(Home Directory)이다. 여기에 기억되는 자료들은 자주 수정되므로 이 등록부의 크기는 계속 늘어 날것이다.

/kernel 이것은 genunix 와 같은 체계를 발생할 때 요구되는 핵심부환경과 2 진파일들을 포함한다.

/lost+found

이것은 잃어 진 파일들의 등록부이다. 여기에 있는 파일들은 리용상태에 있지만 등록부와는 무관계하다. 이 파일들은 원칙적으로 디스크의 물리적정보와 론리적등록부사이의 련결을 끊어 버리는 체계제거의 결과로 "잃어 진"것들이다. Fsck 프로그람은 체계가 발생될 때에 수행되는데 이러한 파일들을 찾아서 lost+found 등록부안에 넣는다.

/mnt 이것은 론리적파일체계에 대한 설치지정용으로 예약되여 있다. 사용자는 /mnt 에 직접 혹은 /mnt 의 보조등록부에 /mnt1, /mnt2, /mnt3 등으로 설치할수 있다.

/net 이것은 원격파일체계에 대한 설치지정용으로 예약되여 있다.

/opt 이것은 응용프로그람들이 설치되는 등록부이다. 응용프로그람판매자들은 자기의 프로그람이 어디에 설치되여야 하는가를 규정하지는 않는다. /opt 아래에는 응용프로그람이 설치되여야 하는 한개의 표준등록부가 있다. 이것은 체계관리자들이 /opt 아래에 적재되여야 할 프로그람들과 프로그람이름들을 예견하도록 할수 있기때문에 체계관리자들에게 편리한 기능으로 된다.

/sbin 이것은 체계발생 혹은 체계완료, 파일체계의 설치에 리용되는 지령들과 원형들을 포함한다. /sbin 은 체계가 발생될 때 사용가능하다. 왜냐하면

체계를 발생할 때 필요한 지령들이 /sbin 에 포함되여 있기때문이다.

/stand 이것은 체계를 발생시킬 때 요구되는 핵심부환경과 2 진파일들을 포함한다. 이 등록부에 포함되는 두개의 중요한 파일들은 system 파 vmunix(핵심부)이다.

/tmp 이것은 임의의 사용자가 림시적으로 파일을 기억시킬수 있는 림시등록부이다. 그러므로 이 등록부에는 중요한 파일을 기억시키지 말아야 한다. 왜냐하면 /tmp 에 기억된 파일은 어떤것이든지 제거될수 있기때문이다. 응용프로그람작업파일들은 /var/tmp 나 /var/opt/appname 에 기억시켜야 한다.

/usr UNIX 조작체계의 대부분은 /usr 안에 포함된다. 이 등록부안에는 지령, 서고, 문서들이 포함된다. /usr 에는 제한된 개수의 보조등록부들이 들어 있다.

/var 이 등록부에는 본질적으로 림시적인 파일들이 포함된다. log 파일과 같은 파일들은 자주 제거되고 수정되는데 이 등록부에 기억된다. 이것은 "변하는" 크기의 등록부라고 생각하면 된다. 응용프로그람이나 지령이 수행될 때 창조하는 log 나 spool 과 같은 파일들은 이 등록부에 들어 간다. 그러나 일부 응용프로그람들은 /var 안에 상태정보를 기억시키기도 한다.

Linux 파일체계구조

Linux 파일체계구조는 이미 언급된 UNIX 변종의 구조와 개념적으로나 실천적으로나 대단히 류사하다. 파일들은 등록부안에 포함되며 등록부들은 임의의 개수의 보조등록부들을 가질수 있다. UNIX 를 포함하여 대부분의 조작체계들은 이런 방법으로 정돈된다.

다음의 실례들은 CalderaLinux 체계의 뿌리준위에 대한 렬거를 보여 주고 있다.

```
# ls -1 /
total 439
-rw-rw-rw-   1   root   root      23   Nov   5   07:10   .lgdb
dr-xr-xr-x   3   root   root    1024   Nov   5   07:10   amd
dr-xr-xr-x   2   root   root     512   Nov   5   07:10   auto
drwxr-xr-x   2   root   root    2048   Nov   5   07:12   bin
drwxr-xr-x   2   root   root    1024   Nov   5   01:44   boot
drwxr-xr-x   2   root   root   10240   Nov   6   22:27   dev
drwxr-xr-x  15   root   root    2048   Nov   7   00:15   etc
drwxr-xr-x   5   root   root    1024   Nov   5   01:43   home
drwxr-xr-x   2   root   root    1024   Nov   5   01:29   initrd
lrwxrwxrwx   1   root   root      11   Nov   5   01:29   install -> /var/lib/LST
```

```
drwxr-xr-x   4    root    root     1024    Nov   5    01:44    lib
drwxr-xr-x   2    root    root    12288    Nov   5    01:29    lost+found
drwxr-xr-x   4    root    root     1024    Nov   5    01:44    mnt
drwxr-xr-x   4    root    root     1024    Sep  10     1996    opt
dr-xr-xr-x   5    root    root        0    Nov   5    01:59    proc
drwxr-xr-x   5    root    root     1024    Nov   6    20:56    root
drwxr-xr--x  2    root    root     2048    Nov   7    00:03    sbin
drwxrwxrwt   6    root    root     1024    Nov   7    00:10    tmp
drwxr-xr-x  21    root    root     1024    Nov   5    01:40    usr
drwxr-xr-x  14    root    root     1024    Nov   5    01:32    var
-rw-r--r--   1    root    root   404158    Nov   5    01:29    vmlinuz
```

몇개의 중요한 등록부이름들은 나타나지 않았지만 대부분의 등록부이름들은 앞에서 서술한것과 같다. Linux 핵심부에서는 vmlinux 가 뿌리등록부에 나타난다. /root 는 사용자 root 의 **홈등록부**(Home Directory)이다. /proc 는 Linux 체계에 대한 정보를 포함하는데 이것은 실지로 등록부처럼 보이는 자료구조정보의 모임이다. 다음의 실례는 /proc 등록부를 현시하고 있다.

```
# ls -1 /Proc
total 0
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   1
dr-xr-xr-x   3     root    root       0   Nov   7   00:16   1011
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   104
dr-xr-xr-x   3      bin    root       0   Nov   7   00:13   106
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   116
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   118
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   146
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   155
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   156
dr-xr-xr-x   3   daemon    root       0   Nov   7   00:13   162
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   167
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   170
dr-xr-xr-x   3     root   65535       0   Nov   7   00:13   178
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   18
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   19
dr-xr-xr-x   3   nobody   65535       0   Nov   7   00:13   193
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   199
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   2
dr-xr-xr-x   3     root    root       0   Nov   7   00:13   20
```

```
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  200
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  201
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  202
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  203
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  204
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  205
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  21
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  273
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  274
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  276
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  3
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  307
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  327
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  358
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  375
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  443
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  444
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  445
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  45
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  482
dr-xr-xr-x  3  root  root         0  Nov  7  00:13  483
dr-xr-xr-x  3  root  root         0  Nov  7  00:16  996
dr-xr-xr-x  3  root  root         0  Nov  7  00:16  997
dr-xr-xr-x  3  root  root         0  Nov  7  00:16  998
-r--r--r--  1  root  root         0  Nov  7  00:13  cmdline
-r--r--r--  1  root  root         0  Nov  7  00:13  cpuinfo
-r--r--r--  1  root  root         0  Nov  7  00:13  devices
-r--r--r--  1  root  root         0  Nov  7  00:13  dma
-r--r--r--  1  root  root         0  Nov  7  00:13  filesys
-r--r--r--  1  root  root         0  Nov  7  00:13  interrupt
-r--r--r--  1  root  root         0  Nov  7  00:13  ioports
-r--------  1  root  root  33558528  Nov  7  00:13  kcore
-r--------  1  root  root         0  Nov  5  07:00  kmsg
-r--r--r--  1  root  root         0  Nov  7  00:13  ksyms
-r--r--r--  1  root  root         0  Nov  7  00:05  loadavg
-r--r--r--  1  root  root         0  Nov  7  00:13  locks
-r--r--r--  1  root  root         0  Nov  7  00:13  mdstat
-r--r--r--  1  root  root         0  Nov  7  00:13  meminfo
```

```
-r--r--r--   1   root   root    0   Nov   7   00:13   modules
-r--r--r--   1   root   root    0   Nov   7   00:13   mounts
dr-xr-xr-x   2   root   root    0   Nov   7   00:13   net
-r--r--r--   1   root   root    0   Nov   7   00:13   pci
dr-xr-xr-x   2   root   root    0   Nov   7   00:13   scsi
lrwxrwxrwx   1   root   root   64   Nov   7   00:13   self -> 110
-r--r--r--   1   root   root    0   Nov   7   00:13   stat
dr-xr-xr-x   5   root   root    0   Nov   7   00:13   sys
-r--r--r--   1   root   root    0   Nov   7   00:13   uptime
-r--r--r--   1   root   root    0   Nov   7   00:13   version
```

우의 실례에서 d 로 시작되는 등록부들은 과제들이다. /proc/kcore 파일은 Linux 체계에 대한 물리적인 기억기를 나타낸다. /proc 에 있는 파일들은 체계에 대한 일부 중요한 정보도 포함하고 있다. 특히 /proc/cpuinfo 파일이 흥미 있는것인데 아래에서는 이 파일에 대한 실례를 제시하고 있다.

```
# more /Proc/cpuinfo
processor          :0
CPU                :686
model              :3
vendor - id        :GenuineIntel
stepping           :4
fdiv bug           :no
hlt - Eug          :no
fpu                :yes
fpu_exception      :yes
cpuid              :yes
WP                 :yes
flags              :fpu vme de pse tsc msr pae mce cx8 11 mtrr pge
mca cmov mmx
bogomips           :266.24
```

more 지령은 다음장에서 고찰하게 된다. 여기서는 이 지령이 파일들을 현시하기 위하여 리용했다는것만 알면 된다.

/boot 등록부는 Linux 핵심부와 LILO boot 관리자가 리용하는 파일들을 포함하고 있다. 다음의 실례는 /boot 를 현시한것이다.

```
# ls -l /boot
total 452
```

```
-rw-r--r--   1  root   root    15954   Feb  11   1997  WHATSIN
-rw-r--r--   1  root   root    15954   Feb  11   1997  WHATSIN-2.0.29-modular
-rw-r--r--   1  root   root      204    Jul  24  19.96  any_b.b
-rw-r--r--   1  root   root      204    Jul  24   1996  any_d.b
-rw-r--r--   1  root   root      512   Nov   5  01:44  boot.0300
-rw-r--r--   1  root   root     4416    Jul  24   1996  boot.b
-rw-r--r--   1  root   root       88    Jul  24   1996  chain.b
-rw-------   1  root   root     7680   Nov   5  01:44  map
-r--r--r--   1  root   root     1565   Mar   7   1997  message
-rw-r--r--   1  root   root      192    Jul  24   1996  os2_d.b
-rw-r--r--   1  root   root   404158   Feb  11   1997  vmlinuz-2.0.29-modular
```

사용자는 체계관리자가 배정해 준 홈등록부와 많이 련관되여 작업한다. 이 등록부는 /home 이다. UNIX 변종들에서 파일체계구조는 류사하지만 사용자는 그 변종들의 자세한 차이도 찾을줄 알아야 하며 UNIX 파일체계구조를 전반적으로 리해해야 한다.

Linux 체계에서 도형대면부를 사용한다면 파일체계를 시각적으로 볼수 있다. 그림 2-2 에서는 Gnome 대면부의 한 부분인 Red Hat Linux 파일관리자를 제시하고 있다.

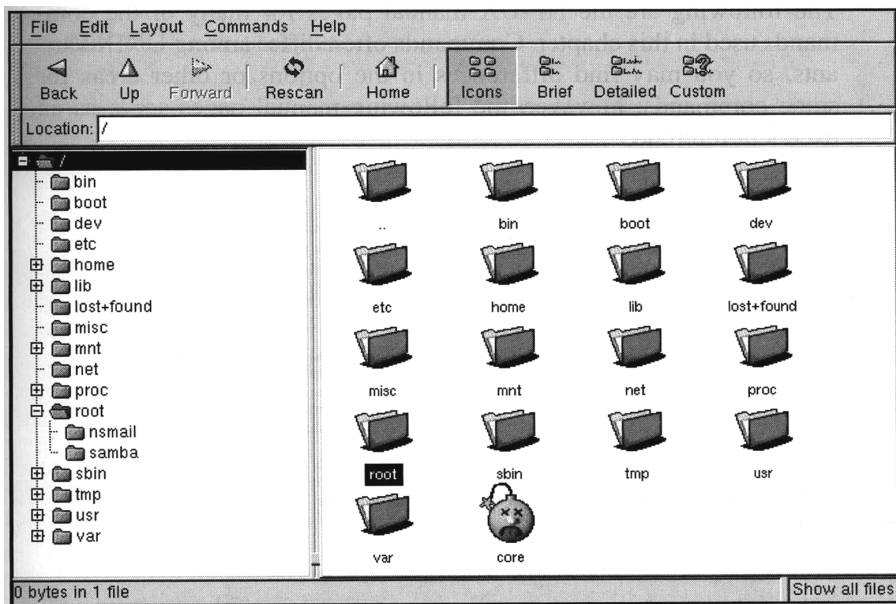



그림 2-2. Red Hat Linux 파일체계를 보여 주는 Gnome 의 파일관리자

그림 2-2 의 왼쪽 부분에는 우리가 앞에서 고찰한 일반적인 UNIX 체계의 등록부들과 같은 등록부들이 많다. 서로 다른 UNIX 변종이라고 하여도 그 변종들은 대부분이 류사한 체계구조를 가진다는것을 볼수 있을것이다.

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령들은 좀 차이나는데 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들이 가장 우수한 기준으로 된다.

file

file – 파일을 분류하기 위하여 검사한다.

---

file(1)

이름

file – 파일종류를 결정한다.

형식

file [-m mfile] [-c] [-f ffile] file …

해설

file 은 매개 파일을 분류하기 위한 검사를 진행한다. 만일 파일이 ASCII 파일로 나타나면 file 은 첫 512byte 를 검사하며 그 언어를 알아 맞춘다. 만일 파일이 실행가능한 a.out 파일이라면 file 은 (0 보다 크다면) 그 파일의 판본번호를 현시한다(ld(1)의 -V 선택항목에 대한 해설을 참고).

file 은 /etc/magic 파일을 리용하여 파일들을 식별한다. 이 파일은 파일종류를 식별하기 위하여 쓰이는 특수한 수값상수 혹은 기호렬상수를 포함하고 있다. /etc/magic 의 첫 시작에서는 설명문형식으로 그 파일의 형식을 설명하고 있다.

선택항목들

file 은 다음의 지령행선택항목들을 인식한다.

-m mfile 식별파일 mfile 을 리용한다.

-c 식별파일의 형식에 오유가 없는가를 검사한다. 이 검사는 대체로 리용하지 않는다. 이 선택항목이 규정되면 파일식별은 진행되지 않는다.

-f ffile ffile 은 식별해야 할 파일들의 목록이다. file 은 ffile 에 있는 매개 파일들을 식별한다.

외부적영향들

환경변수들

LC_MESSAGES 는 현시되는 통보의 언어를 결정한다. 만일 LC_MESSAGES 가 환경에서 규정되지 않았거나 그것이 빈 기호렬로 설정되여 있다면 LANG 의 값은 기정값으로 사용된다. LANG 이 규정되지 않았거나 빈 기호렬로 설정되여 있다면 LANG 대신에 "C"가 표준적으로 사용된다(lang(5)를 참고).
만일 어떤 **국제화변수**(internationalized variable)가 무효하게 설정된 값을 포함하고 있으면 file 은 모든 국제화변수들을 "C"로 설정한것처럼 동작한다(environ(5)을 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드가 보장된다. 그러나 ASCII 본문파일이 아닌 파일들은 모두 자료로 식별된다.

관련항목

ld(1)

표준일치

file:SVID2, SVID3, XPG2, XPG4

ls

ls – 등록부의 내용을 현시한다.

---

ls(1)

이름

ls, l, ll, lsf, lsr, lsx – 등록부내용을 현시한다.

형식

ls [-abcdefgilmnopqrstuxACFLR1] [names]
l [ls-options] [names]
ll [ls-options] [names]
lsf [ls-options] [names]
lsr [ls-options] [names]
lsx [ls-options] [names]

해설

매개 등록부에 대하여 ls 지령은 등록부의 내용을 현시한다. 매 등록부인수에 대하여 ls 는 그안에 있는 대상들의 이름과 기타 필요한 정보를 반복한다. 현시하는 내용은 증가순서로 표준적으로 정돈된다(아래의 환경변수항목을 참고). 아무런 인수도 규정하지 않으면 현재등록부가 현시된다. 여러개의 인수가 주어 지면 그 인수들이 먼저 정돈된다. 이때 파일인수들은 등록부들과 그 등록부들의 내용보다 먼저 놓이도록 정돈된다. 만일 특권을 가진 사용자라면 거의 모든 파일이 기정으로 현시된다.

3 개의 기본적인 출력형식이 있다. 선택된 형식은 출력이 login 장치에서 가지는가 가지지 않는가에 의존한다(출력장치파일이 tty 장치인가 아닌가에 따라 결정된다.). 또한 출력형식은 선택항목으로도 조종할수 있다. 등록장치에 대한 기정형식은 등록부의 내용을 다중렬형식으로 출력하는것인데 매개 구체례는 렬별로 수직방향으로 정돈된다(등록부이름이 아니라 개별적인 파일이름들이 인수로 되여 있을 때 그 파일이름들은 항상 렬안에서 페지의 가로방향으로 정돈된다. 왜냐하면 개별적인 파일이름들의 길이가 제한이 없기때문이다.).

표준출력장치가 login 장치가 아닐 때 기정형식은 한행에 한개 구체례를 현시하는것이다. -C 및 -x 선택항목들은 다중렬형식을 취할수 있게 하며 -m 선택항목은 매 파일들이 페지에서 가로방향으로 반점으로 구분되면서 현시되는 흐름식출력을 할수 있게 한다. -C, -x, -m 선택항목에 대한 출력형식을 규정하기 위하여 ls 는 환경변수인

COLUMNS 를 사용하며 매개 출력행에서 가능한 기호의 개수를 결정한다. 이 변수가 설정되지 않으면 terminfo 자료기지를 리용하여 환경변수 TERM 에 기초한 렬 개수를 결정한다. 이 정보를 얻을수 없으면 80 개 렬로 가정되게 된다.

선택항목들

ls 는 다음의 선택항목들을 인식한다.

-a 모든 구체례들을 현시한다. 보통 점(.)으로 시작되는 구체례는 현시하지 않는다.

-b 도형이 아닌 기호들이 8 진식\ddd 표시법으로 인쇄되게 한다.

-c 색인마디의 최종수정시간(파일이 창조되거나 변경된것 등)을 리용하여 정돈(-t)하거나 인쇄(-l(ell))한다.

-d 인수가 등록부일 때 그것의 내용이 없이 이름만 현시한다. 등록부의 상태를 얻기 위하여 -l(ell)과 함께 자주 리용된다.

-e 파일의 확장된 속성들을 인쇄한다. 파일이 확장속성을 가지면 이 선택항목은 확장크기, 예약공간, 배정기발들을 출력한다. 이 선택항목은 -l(ell)선택항목과 함께 리용되여야 한다.

-f 매 인수가 등록부라고 인식하며 매 **슬로트**(slot)에서 찾은 이름을 현시한다. 이 선택항목은 -l(ell), -t, -s, -r 를 무효하게 하며 -a 를 유효하게 한다. 순서는 매 구체례가 등록부에서 나타나는 순서이다.

-g 소유자이름은 생략되며 그룹이름이 인쇄된다는것을 내놓고는 -l(ell)과 동일하다. -l(ell) 및 -g 가 둘 다 규정되면 소유자이름이 인쇄되지 않는다.

-i 매 파일의 색인마디번호를 첫렬에 현시한다. 다중렬형식이 사용될 때 이 번호는 파일이름앞에 놓인다.

-l(ell) 련결개수, 소유자, 그룹, 바이트수, 최종수정시간 등의 긴 형식의 출력을 진행한다(아래의 해설항목과 접근조종목록항목을 참고). 최종접근시간이 6 개월이전이거나 미래의 시간이라면 년도는 현재 접근시간의 년도로 바뀐다. 파일이 특수파일이라면 크기마당은 크기값이 아니라 장치번호를 포함한다.

-m 흐름식출력형식

-n 소유자의 UID 및 그룹의 GID 번호들이 인쇄된다는것을 제외하고는 -l(ell)과 동일하다.

-o 그룹이름이 아니라 소유자이름만이 인쇄된다는것을 제외하고는 -l(ell)과 동일하다(-l(ell) 및 -o 가 둘 다 규정되면 그룹이름은 인쇄되지 않는다.).

-p 파일이 등록부라면 그 파일뒤에 빗선(/)을 붙인다.

-q 파일이름안의 도형이 아닌 기호들은 (?)으로 인쇄되게 한다.

-r 정돈순서를 거꿀순서로 정한다.

-s 매 구체례에 대하여 간접적블로크를 포함하여 블로크의 크기를 제시해 준다. 인쇄되는 첫번째 구체례는 등록부에서 블로크의 총 개수이다. 다중렬형식의 출력에서 블로크의 개수는 파일이름 앞에 놓인다.

-t 최종수정시간을 먼저 놓으면서 수정시간에 따라 정돈한다.

-u 최종수정시간대신에 최종접근시간을 리용하여 정돈(-t 선택항목) 하거나 인쇄(-l(ell)선택항목)한다.

-x 페지에서 구체례들을 가로방향으로 정돈하여 다중렬형식으로 출력하도록 한다.

-A 현재등록부 "."와 어미등록부 ".."가 현시되지 않는다는것을 제외하고는 -a 와 동일하다. 특수권한을 가진 사용자인 경우에 이 기발값은 기정으로 on이고 -A 에 의하여 "." 및 ".."이 잘리운다.

-C 렬별로 정돈하여 여러개 렬로 출력한다.

-F 파일이 등록부이거나 어떤 등록부에 대한 기호적련결이라면 그 파일이름뒤에 빗선(/)을 붙인다. 파일이 실행파일이면 *을, 파일에 대한 기호적련결이면 @, FIFO 이면 수직막대기 ｜를 각각 붙인다.

-L 인수가 기호적련결이면 련결 그자체가 아니라 그 련결이 가리키는 파일이나 등록부를 현시한다.

-R 보조등록부들에 대하여 재귀적으로 현시한다.

-1(하나) 파일이름들은 출력장치에 관계없이 하나의 렬로 된 출력형식으로 현시한다.

다음과 같은 호상 배타적인 쌍들에서 한개이상의 선택항목들을 규정해도 오유는 발생하지 않는다.

-C 와 -l(ell), -m 와 -l(ell), -x 와 -l(ell), -C 와 ?l(하나), -c 와 ?u.

ls 가 제공하는 속기형식은 다음과 같다.

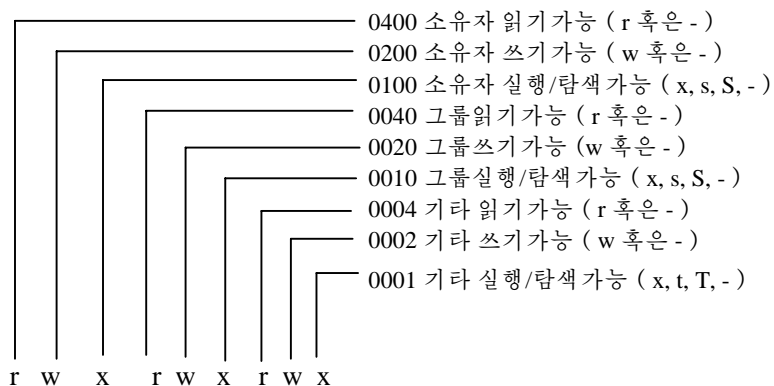
l	ls -m 과 동일
ll	ls -l(ell)과 동일
lsf	ls -F 와 동일
lsr	ls -R 와 동일
lsx	ls -x 와 동일

속기형식의 표기법은 ls 에 대한 련결에 의하여 실현된다. 속기형식의 선택항목들은 정확히 긴 형식의 선택항목처럼 동작한다.

방식비트의 해석(-l 선택항목)

-l(ell)선택항목에 의하여 만들어 지면서 출력에서 현시되는 매 방식은 10 개의 기호들로 구성된다. 첫 기호는 다음과 같이 구체례의 종류를 지적한다.

d	등록부
b	블로크파일
c	기호파일
l	기호적련결
p	FIFO(**파이프**(Pipe)라고도 부른다.)
n	망파일
s	소케트
-	일반파일

다음의 9 개 기호들은 3 개 비트씩 세개의 조로 해석되는데 매개 조는 소유자, 그룹, 기타 사람들에 대한 접근허락을 규정한다.
.방식기호들은 다음과 같이 해석된다.

-	대응하는 허락을 보장하지 않는다.
r	대응하는 사용자클라스에 읽기허락을 제공한다.
w	대응하는 사용자클라스에 쓰기허락을 제공한다.
x	대응하는 사용자클라스에게 실행/탐색허락을 제공한다.
s	대응하는 사용자클라스에게 실행(탐색)허락을 제공하며 추가적으로 소유자에게는 SUID(Set User ID)허락을, 그룹에는 SGID(Set Group ID)허락을 제공한다.
S	대응하는 사용자클라스에게 실행(탐색)허락을 제공하지 않는다는것을 제외하고 s와 동일하다.
t	마지막조에서만 쓰인다. 실행(탐색)허락이 기타 사용자들에게 제공되며 "**보호비트**(Sticky Bit)"가 설정된다. chmod(2)의 S_ISVTX해설항목을 참고.
T	실행(탐색)허락이 기타 사용자들에게 제공되지 않는다는것을 제외하고는 t와 동일하다.

등록부나 파일의 크기가 바이트 혹은 블로크단위(-s 혹은 -l(ell))로 표시되도록 하는 선택항목이 규정되였을 때 간접블로크까지 포함하는 블로크의 총수도 렬거되는 목록앞에 인쇄된다.

접근조종목록(ACL)

ACL구체례를 가지는 파일이 있다면 선택항목 -l(ell)은 파일의 허락뒤에 더하기기호(+)를 현시한다. 나타나는 허락들은 파일의 접근조종목록에 대한 개괄된 허락이다. 이것들은 st_mode마당의 stat( )에 의하여 귀환되는것이다(stat(2)를 참고). 접근조종목록의 내용을 현시하려면 lsacl지령을 사용한다(lsacl(1)과 acl(5)을 참고).

외부적영향

환경변수들

COLUMNS변수가 설정되였으면 ls는 렬출력을 지적하면서 결정된 너비를 리용한다.

LANG 은 LC_ALL 및 대응하는 환경변수(LC_로 시작된다.) 둘 다 지역(locale)을 규정하지 않았을 때 지역분류를 위하여 리용하는 지역을 결정한다. LANG 이 설정되지 않았거나 빈 기호렬로 설정되였다면 기정값으로는 "C"가 리용된다(lang(5)를 참고).

LC_COLLATE 는 출력이 정돈되는 순서를 결정한다.
LC_CTYPE 는 선택항목 -b 와 -q 에 대하여 어느 기호가 비도형기호로 분류되는가(파일이름안의 어느 기호가), 단일바이트기호 혹은 다중바이트기호로 해석되는가를 결정한다.

LC_TIME 은 -g, -l(ell), -n, -o 선택항목에 의한 날자 및 시간출력형식을 결정한다.

LC_MESSAGE 는 (날자 및 시간이 아닌) 통보기호렬의 언어를 결정한다.

만일 어떤 국제화변수가 무효한 설정을 포함한다면 ls 는 모든 국제화변수가 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임의 보장
단일 및 다중바이트기호코드모임이 보장된다.

귀환값
ls 는 다음의것들중에서 어느 한 값으로 탈퇴한다.

0	모든 입력파일들이 성공적으로 현시되였다.
>0	파일에로 접근할 때 오유로 하여 ls 는 실패하였다.

오유를 일으키는 조건은 다음과 같다.

- 규정된 파일을 찾지 못하였다.
- 사용자는 등록부를 읽을수 있는 허락을 가지고 있지 못하다.
- 과제수행을 위한 충분한 기억이 없다.
- 무효한 선택항목이 규정되여 있다.

실례
현재작업등록부의 파일들을(파일크기까지 포함하여) 모두 렬거해 보자. 이때 최근에 수정된 파일이 제일 앞에 놓이도록 하고 .으로 시작된 파일들도 출력되도

록 한다.

ls - alst

경고

출력장치가 login 장치(tty)인가 혹은 다른 장치인가에 따라 선택항목을 설정할 때 ls -s 와 ls -s | lp 처럼 많이 차이난다. 한편 이 설정을 리용하지 않으면 ls 를 리용한 이전의 쉘스크립트는 거의나 불가피하게 실패하게 된다.

파일이름에서 인쇄불가능한 기호들은 렬형식의 출력선택항목들을 혼돈시킬수 있다.

종속성

NFS -l(ell)선택항목은 선택적인 접근조종목록구체례들의 존재를 나타내기 위하여 망파일들의 접근허락비트들뒤에 더하기기호(+)를 현시하지 않는다.

저자

ls 는 AT&T, 캘리포니아종합대학, 버클리와 HP 에 의하여 개발되였다.

관련항목

chmod(1), find(1), lsacl(1), stat(2), acl(5)

표준일치

ls: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제3장. 파일과 등록부에 대한 작업 - 허락, 지령, 파일이름확장, 통용기호

이 장에서는 UNIX 파일체계구조와 관련되는 여러가지 문제들과 일상적으로 파일체계를 리용하기 위한 방법에 대하여 고찰한다. 이 장에서 서술하는 내용은 다음과 같다.

- 허락
- 절대경로이름과 상대경로이름
- 파일이름확장과 통용기호
- pwd, cd, chmod, cp, mv, mkdir, rm, rmdir 지령들
- 여러 지령들을 사용한 실례

허　락

허락은 등록부내용을 현시하는 ls 지령에서 고찰하는것이 제일 좋다. **허락**(Permission)은 UNIX 체계에서 파일 및 등록부가 어떻게 보호되는가를 규정하는 수단이다.

UNIX 는 다중사용자체계이기때문에 많은 사용자들이 체계의 파일들을 접근할수 있다. 허락은 누가 어떤 파일들을 접근할 자격이 있는가 하는것을 조종한다.

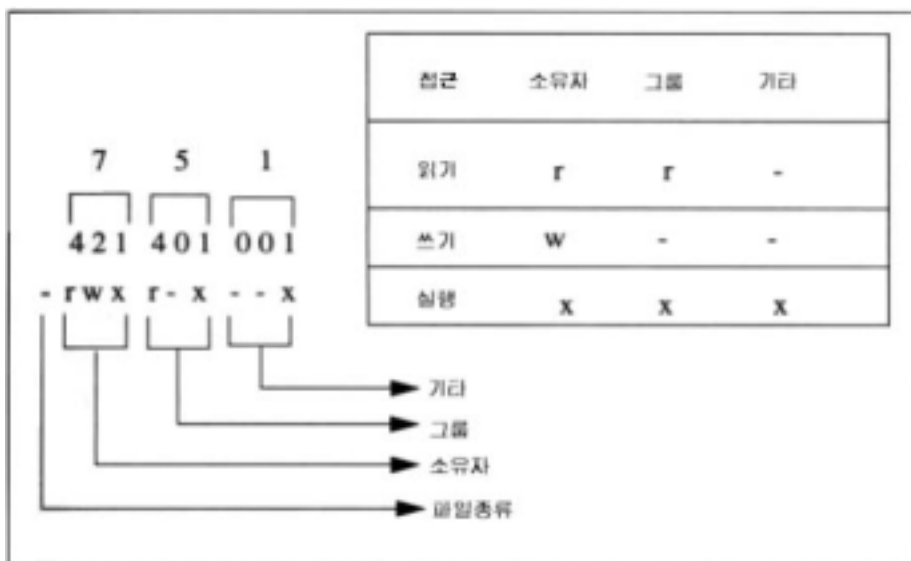



그림 3-1. 파일 sort 의 허락

다음의 실례는 ls -l지령의 결과를 보여 주고 있다.

$ ls -l sort
-rwxr-x--x 1 marty users 120 Jul 26 10∶20 sort

이 지령은 sort 파일에 대한 많은 정보를 생성하였다. 생성된 정보가 어떤 정보인가에 대하여 해석하여 보자. 현시된 내용에서 제일 처음으로 나타난 문자들의 한개조(-rwxr-x--x)는 그림 3-1에서 보여 준것처럼 4개의 마당으로 구별할수 있다.

첫 문자는 파일종류를 나타낸다. ls -l 지령에서 현시하는 파일종류들은 이미 앞장에서 언급한 파일종류와 동일한 관점에서 해석되는것이 아니다. 이 지령에서 현시되는 파일종류들은 그림 3-2에서 보여 주고 있다.

UNIX 변종들에서 파일종류들은 약간씩 변하지만 그림 3-2 에서 보여 준 파일종류들은 대부분의 UNIX 변종들에서 공통적인것들이다.

첫 문자	파일종류
−	본문파일과 같은 **일반파일** (Ordinary File)
b	블로크장치파일
c	문자장치파일
d	등록부
l	련결
n	망파일

그림 3-2. ls지령의 파일종류

체계의 매 파일에 대하여 UNIX들은 3가지 **접근클라스**(class of access)를 제공한다.

- 사용자접근(u). 파일소유자에게 제공되는 접근
- 그룹접근(g). 파일소유자와 동일한 그룹의 성원들에게 제공되는 접근
- 기타접근(o). 그밖의 사용자들에게 제공되는 접근

이러한 접근허락은 등록부내용을 길게 현시하는 ls -l 과 같은 지령이 수행될 때 읽기(r), 쓰기(w), 실행(x)의 위치에서 정의된다.

표 3-1 sort파일에 대한 허락

접근	사용자접근	그룹접근	기타
읽기	r	r	-
쓰기	w	-	-
실행	x	x	x

"-"로 나타나는 허락은 접근허락이 제공되지 않는다는것이다. 이외에도 s, S, t, T와 같은 허락이 더 있지만 여기서는 고찰하지 않기로 한다.

다음으로 3개의 조로 구성되는 허락들은 접근준위를 가지고 접근허락을 규정하고 있다. 우의 실례에서는 소유자 marty가 파일의 읽기, 쓰기, 실행허락을 가지며 users그룹에 속하여 있는 성원들은 읽기와 실행접근이 허용되여 있다. 그리고 기타 사용자들은 실행접근만을 할수 있다.

읽기, 쓰기, 실행에 대한 정의는 파일과 등록부에서 약간 차이난다. 파일에 대한 읽기, 쓰기, 실행의 정의는 다음과 같다.

읽기: 파일을 읽을수 있는 허락

쓰기: 파일을 변경시키거나 쓸수 있는 허락

실행: 프로그람을 실행시킬수 있는 허락

등록부에 대한 읽기, 쓰기, 실행의 정의는 다음과 같다.

읽기: 등록부의 내용을 현시할수 있는 허락

쓰기: 등록부에 파일을 창조, 제거하거나 보조등록부를 창조할수 있는 허락

실행: cd지령을 리용하여 등록부를 변경시킬수 있는 허락

chmod지령을 고찰할 때에 허락에 대하여 다시 보기로 한다.

절대 및 상대경로이름

이미 절대경로와 상대경로로 볼수 있는 두가지 경우를 고찰하였다. 그 하나는 체계적관점에서 중요한 등록부들이고 다른 하나는 가입된 사용자의 등록부들이다.

제2장에서 우리는 UNIX파일체계가 계층화되여 있으며 그 체계의 제일 우에는 뿌리등록부가 있고 뿌리등록부아래에 파일 및 등록부들이 있다는것을 보았다. 이러한 계층나무를 **절대경로이름**(Absolute Path Name)과 **상대경로이름**(Relative Path Name)이라는 두가지 수단에 의하여 탐색할수 있다.

그림 3-3에 있는 denise사용자는 많은 파일들을 가지고 있다. 사용자는 파일들을 창조할수 있으며 체계관리자는 사용자가 체계에 가입한후에 그 사용자의 환경을 정의하기 위한 여러개의 표준파일들을 제공해 준다. denise는 자기의 사용자령역에 파일과 등록부들을 많이 가질수 있다.

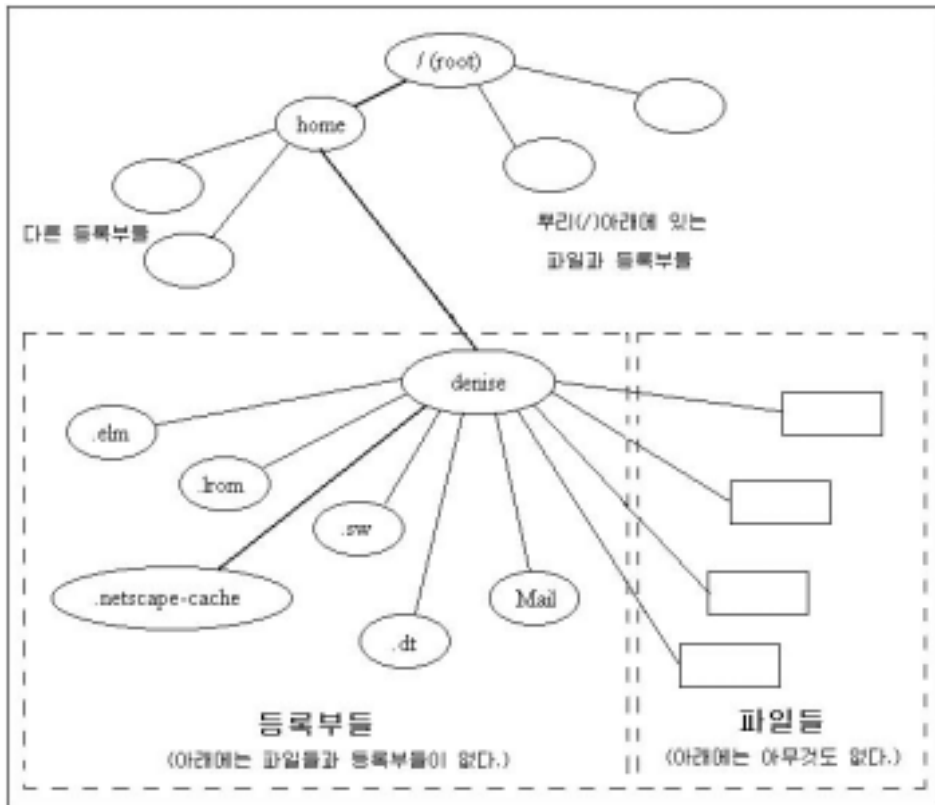



그림 3-3. denise 의 홈등록부

대체로 사용자들은 /home 등록부아래에 자기의 홈등록부들을 가진다. 만일 절대경로이름을 리용하여 denise 의 어떤 보조등록부에로 가자면 계층나무에서 경로를 완전하게 규정하여 탐색해야 한다. denise 아래에 있는 .netscape-cache 등록부에 가자면 그림 3-4 에서처럼 절대경로를 보아야 한다.

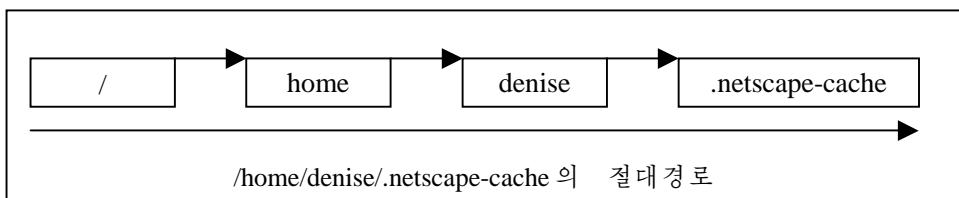



그림 3-4. 절대경로

즉 뿌리(/)로부터 home, denise, .netscape-cache 로 가야 한다. 등록부변경지령(cd)는 다음과 같이 쓸수 있다.

$ cd /home/denise/. netscape-cache

이것은 뿌리로부터 시작하여 계층나무를 따라 전진했기때문에 절대경로이다. 절대경로는 현재 어느 등록부에서 작업하고 있는가에는 관계없이 지정할수 있지만 자기가 접근하려는 파일이나 등록부의 가까이에 있는 경우에는 절대경로지정이 오히려 시간랑비로 된다. 실례로 사용자가 이미 /home/denise 에서 작업하고 있었다면 다음의 상대경로이름을 리용하여 .netscape-cache 로 쉽게 갈수 있다.

$ **cd .netscape-cache**

상대경로이름은 계층나무의 제일 우에까지 되돌아 가서 뿌리(/)로부터 출발하지 않기때문에 보다 더 짧다. 상대경로는 /home/denise 와 같은 파일체계계층의 일정한 지점에서 출발하여 .netscape-cache 와 같은 상대적인 경로에 입장한다.

파일이름확장과 통용기호

파일체계관련의 지령들을 고찰하기에 앞서 파일이름확장에 대하여 론의하는것이 중요하다. 표 3-2 는 공통적인 몇가지 파일이름확장과 **패턴정합**(Pattern Matching)을 보여 주고 있다.

표 3-2 파일이름확장과 패턴정합

문 자	실 례	해 설
*	1) **ls *.c**	0 개이상의 기호들과 맞춘다.
?	2) **ls conf.?**	임의의 한개 기호와 맞춘다.
[목록]	3) **ls conf.[co]**	목록안의 임의의 기호와 맞춘다.
[아래 한계-웃 한계]	4) **ls libdd. 9873[5-6].sl**	범위내의 임의의 기호와 맞춘다.
문자렬{문자렬 1, 문자렬 2, …}	5) **ls ux*.{700, 300}**	기호렬을 { }안의 내용으로 확장
~	6) **ls -a ~**	홈등록부
~사용자이름	7) **ls -a~gene**	사용자이름의 홈등록부

표 3-2에서 보여 준 실례들을 구체적으로 설명하기로 하자.

① ".c"로 끝나는 파일들을 모두 보기 위해서는 다음과 같이 써야 한다.

$ **ls *.c**

conf. SAM.c conf.c

② 확장자가 한개 기호이고 이름이 "conf"인 모든 파일을 보려면 다음과 같이

쓸수 있다.

```
$ ls conf.?
        conf.c    conf.o          conf.l
```

③ 이름이 "conf"이고 확장자가 "c" 혹은 "o"로 되여 있는 모든 파일을 보려면 다음과 같이 써야 한다.

```
$ ls conf.{co}
        conf.c    conf.o
```

④ 일정한 범위안에 있는 류사한 이름을 가진 파일들을 보기 위해서는 다음과 같이 쓸수 있다.

```
$ ls libdd9873[5-6]
        libdd98735.sl          libdd98736.sl
```

⑤ 이름이 "ux"로 시작되고 확장자가 "300" 또는 "700"인 파일들을 보기 위해서는 다음과 같이 써야 한다.

```
$ ls ux*.{700, 300}
        uxbootlf.700           uxinstfs.300
```

⑥ 홈등록부에 있는 파일들을 보기 위해서는 ~기호를 사용하여야 한다.

```
$ ls -a ~
        .            .cshrc.org    .login        .shrc.org
        ..           .exrc         .login.org    .cshrc
        .history     .profile
```

⑦ 사용자의 홈등록부에 있는 파일들을 보려면 다음과 같이 써야 한다.

```
$ ls -a ~gene
        .            .history      splinedat     under.des
        ..           .login        trail.txt     xtra.part
        .chsrc       .login.org    ESP-File
        .chsrc.org   .profile      Mail
        .exrc        .shrc.org     opt
```

pwd와 cd

절대경로이름과 상대경로이름을 고찰하면서 앞에서 등록부를 변경시키기 위하여 cd 지령을 사용하였다. 계층화된 파일체계의 임의의 위치에서 목적하는 등록부에 대한 변경허락을 가지고 있다면 그 등록부를 cd지령으로 변경시킬수 있다. 다음의 실례에서는 절대경로이름을 리용하여 등록부를 변경시키는것을 보여 주고 있다.

$ **cd /home/denise/. netscape-cache**

이 지령은 계층화된 파일체계의 현재위치에는 관계없이 등록부를 /home/denise /. netscape-cache 로 변경시킨다. 그러나 계층화된 파일체계에서 현재의 위치가 /home /denise 일 때에는 다음의 실례에서와 같이 상대경로이름을 리용하여 등록부를 .net scape-cache 로 변경시킬수 있다.

$ **cd . netscape-cache**

현재의 위치로부터 파일체계의 상대적위치로 등록부를 변경시키기 위해서는 현재위치를 판정할 필요가 있다. pwd 지령은 **현재작업등록부**(Present Working Directory)를 귀환하는 지령이다. 앞의 실례에서는 먼저 pwd 지령을 리용하여 현재의 위치를 다음과 같이 판정할수 있다.

$ **pwd**
/home/denise
$ **cd . netscape-cache**
$ **pwd**
/home/denise/. netscape-cache
$

다음으로 두개의 점(..)으로 그 웃준위에 있는 등록부에로 이동하는 방법을 보자.

$ **pwd**
/home/denise/. netscape-cache
$ **cd ..**
$ **pwd**
/home/denise
$

실례에서 보여 준바와 같이 두개의 점은 현재작업등록부의 어미등록부로 이동시킨다. 자기의 홈등록부로 돌아 오자면 다음실례에서 보여 준것처럼 아무런 인수도 없이 cd지령을 리용하면 된다.

$ **pwd**
/tmp
$ **cd**
$ **pwd**
/home/denise
$

이것은 현재의 위치가 계층구조의 어디에 있든지 관계없이 자기의 홈등록부로 즉시에 돌아 올수 있다는것을 보여 주고 있다. 지금까지는 **쉘파라메터**(Shell Parameter)를 리용하지 않았는데 다음의 실례에서는 자기의 홈위치를 정의하는 쉘파라메터($HOME)를 사용하는것을 보여 주고 있다.

$ **pwd**
/tmp
$ **cd $HOME**
$ **pwd**
/home/denise
$

이처럼 pwd 지령을 사용하면 언제나 자기의 현재등록부를 얻을수 있으며 cd 지령을 사용하면 임의의 등록부로 변경시킬수 있다.

cd - 현재의 등록부를 변경시킨다.

인수들

없음	인수가 하나도 없으면 홈등록부로 변경한다. 이것은 HOME 환경변수에 의하여 정의된다.
..	두개의 점표시는 현재등록부를 어미등록부로 이동시킨다.
경로	변경시키려는 절대경로 혹은 상대경로를 규정할수 있다.

pwd - 현재작업등록부를 귀환한다.

실례

$ **pwd**
/home/denise/. netscape-cache
$ **cd ..**
$ **pwd**
/home/denise
$

chmod

chmod 지령은 파일에 대한 허락을 변경하기 위하여 사용하여야 한다. 앞에서 보여 준 sort 파일을 리용하여 chmod를 고찰하자.

```
$ ls -1 sort
-rwxr-x--x    1 marty    users   120 Jul 26 10:20 sort
```

그림 3-5는 파일 sort에 대한 허락의 내용을 보여 주고 있다.

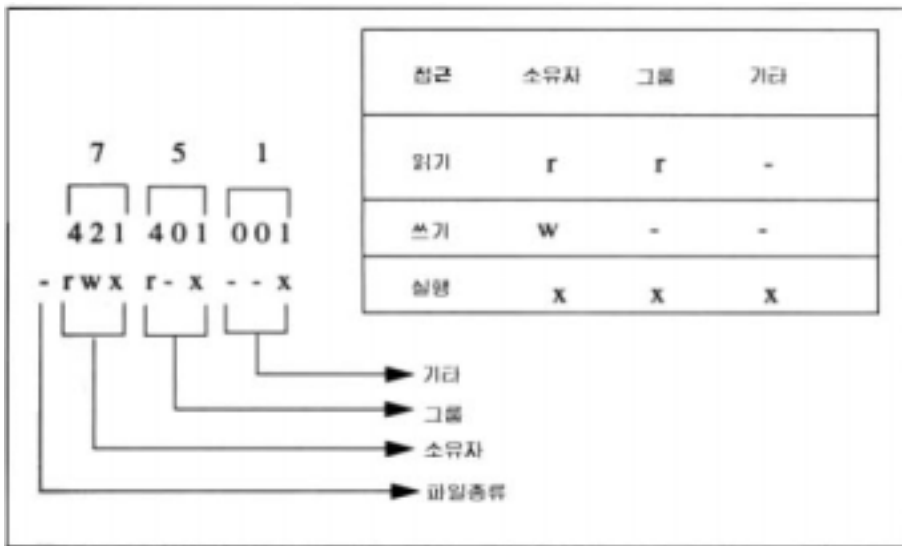



그림 3-5. 파일 sort의 허락들

파일의 종류는 거의나 변경할수 없다. 그러나 그 파일의 허락은 조종할수 있다. chmod 지령은 파일이나 등록부의 허락을 변경하는 지령이다. 파일의 소유자에게는 그 파일의 허락을 변경시킬 필요가 있을수 있다. 보통 허락변경은 기호나 수자로 한다. 여기서는 **수값방식**(Numeric Mode)을 보기로 하자. **기호방식**(Symbolic Mode)에서 사용하는 기호의 의미는 뒤에 있는 chmod의 개요에서 서술하고 있다.

그림 3-5에서 볼수 있는것처럼 sort의 허락은 751이다. 여기서 7은 소유자용(백의 자리수), 5는 그룹용(열의 자리수), 1은 기타용(하나의 자리수)이다. 그림 3-6에서는 매 위치의 의미를 보여 주고 있다.

사용자는 소유자, 그룹, 기타에 대한 허락을 선택하여 chmod 지령으로 파일이나 등록부에 그 허락을 배정할수 있다. 쓰기허락과 같은 일부 허락들은 드물게 사용된다. 그러나 그림 3-6에서는 가능한 모든 경우를 보여 주고 있다. sort 파일에 대하여 그룹의 쓰기허락을 추가하고 기타 사용자들에 대한 모든 허락을 제거하려면 chmod 지령에서 단순히 해당한 수값을 지적하면 된다.

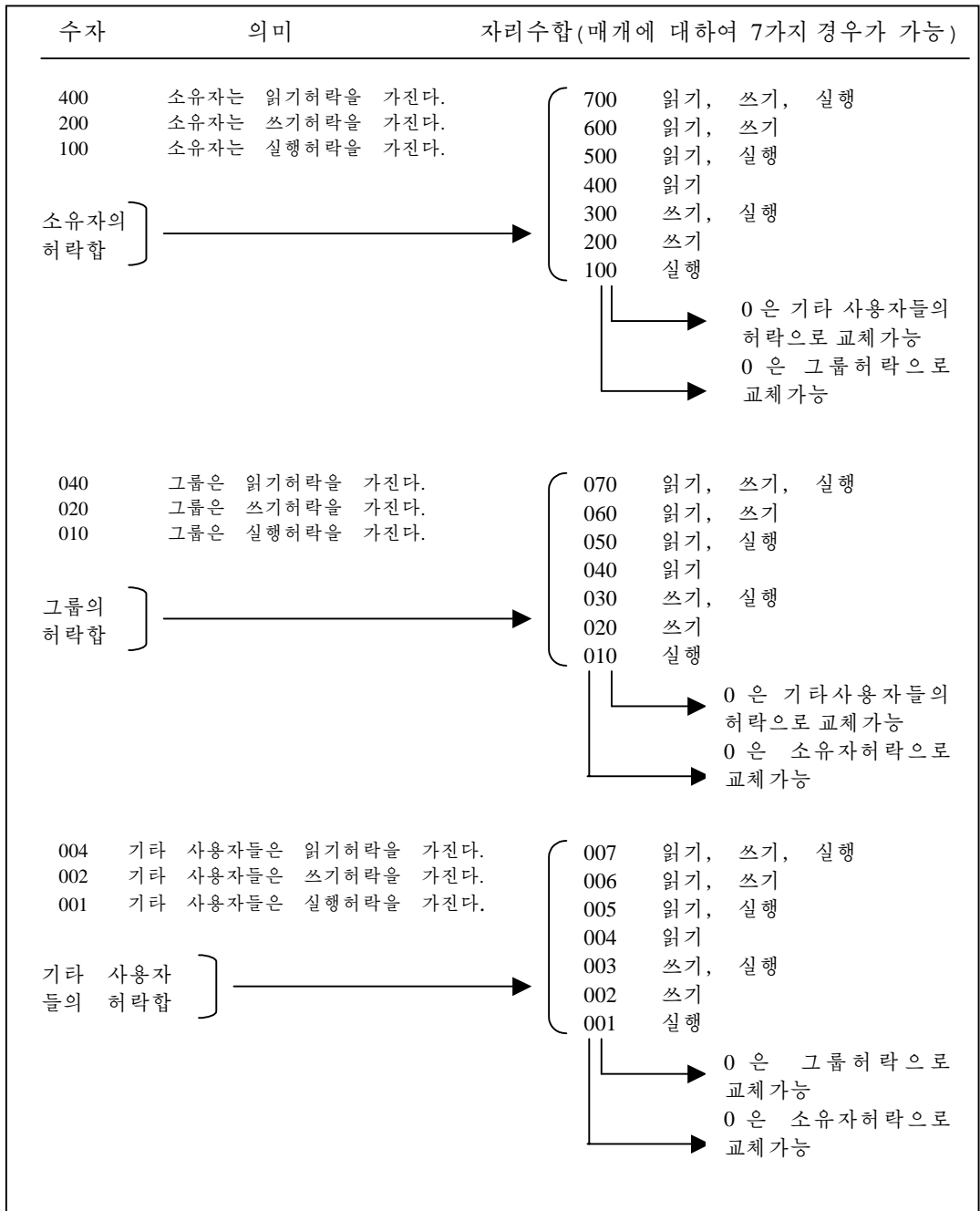



그림 3-6. 수자허락들의 개괄

다음의 실례에서는 먼저 sort 파일의 허락들을 렬거하고 그다음 그것의 허락들을 변경한 다음 변경된 sort 의 허락들을 다시 렬거하도록 하고 있다.

```
$ ls -l sort
-rwxr-x--x   1   marty   users   120   Jul   26   10:20   sort

$ chmod 770 sort
$ ls -l sort
-rwxrwx---   1   marty   users   120   Jul   26   10:20   sort
```

기호방식으로 우와 같이 허락들을 변경하려면 다음과 같이 한다.

```
$ ls -l sort
-rwxr-x--x   1   marty   users   120   Jul   26   10:20   sort

$ chmod g+w, o-x sort

$ ls -1 sort
-rwxrwx---   1   marty   users   120   Jul   26   10:20   sort
```

기호방식에서 chmod 지령은 소유자는 u, 그룹은 g, 기타는 o, 모두는 a 로 표시하며 허락의 추가는 +, 제거는 -, 교체는 =로 표시한다. 또한 읽기, 쓰기, 실행허락은 각각 r, w, x 로 표시한다. 앞의 실례에서 그룹(g)에 대한 쓰기허락(w)은 추가(+)되며 기타(o)에 대한 실행허락(x)은 제거(-)된다. 아래에서 chmod 의 기호들을 개괄한다.

chmod - 다음의 기호방식을 리용하여 규정된 파일의 허락을 변경한다.

사용자구분

u	소유자
g	그룹
o	기타
a	모두

수행연산

+	허락추가
-	허락제거
=	허락교체

허락규정

r	읽기
w	쓰기

x	실행
u	사용자허락복사
g	그룹허락복사
o	기타 허락복사

cp

cp 지령은 파일을 한 장소로부터 다른 장소로 복사할 때 리용된다. 일반파일은 변경되지 않는다. 파일을 복사하려는 목적등록부에 대한 접근허락을 가지고 있으면 그 등록부에 요구대로 파일들을 복사할수 있다.

아래에서는 cp 로 수행할수 있는 복사형태들을 서술하고 있다. 제 2 장에서 고찰한 바와 같이 UNIX 에는 많은 파일종류가 있기때문에 원천 및 목적파일에 대한 경로를 여러가지 방법으로 규정할수 있다.

- 원천파일을 새로운 파일이름으로 복사한다.
- 여러개 원천파일을 다른 등록부에 복사한다.
- 여러개 원천파일을 동일한 등록부에 복사한다.
- 등록부전체를 다른 등록부에 복사한다.
- 여러개 등록부들을 다른 등록부들에 복사한다.

다음의 실례는 하나의 파일을 동일한 등록부안에서 새로운 파일이름으로 복사한다.

$ **cp krsort krsort.sav**

만일 krsort.sav 라는 파일이 이미 존재한다면 어떤 현상이 일어 나겠는가? 파일이 이미 존재한다면 그 파일은 복사되는 새로운 krsort.sav 파일로 교체된다. 이러한 겹친 현상을 막기 위하여 cp 지령에 -i 를 리용하면 복사하기전에 겹치게 하겠는가를 물어 본다. 사용자의 대답이 확정적이라면 낡은 파일은 새로운 파일로 겹쳐 진다.

다음의 실례는 먼저 등록부의 내용을 렬거하고 -i 와 함께 cp 지령을 리용하여 krsort.c 파일을 이미 존재하는 krsortorig.c 로 복사한다. krsortorig.c 를 겹치게 하겠는가고 물어 볼 때 n 으로 대답함으로써 그 파일이 겹쳐 지지 않고 아무런 복사도 하지 않게 한다.

```
$ ls -l
total  168
-rwxr-xr-x   1   denise   users   34592   Oct   31   11:27   krsort
-rwxr-xr-x   1   denise   users    3234   Oct   31   11:27   krsort.c
-rwxr-xr-x   1   denise   users   32756   Oct   31   11:27   krsort.dos
```

```
-rw-r--r--   1  denise  users   9922  Oct  31  11:27  krsort.q
-rwxr-xr-x   1  denise  users   3085  Oct  31  11:27  krsortorig.c
```
$ **cp -i krsort.c kroortorig.c**
overwrite krsortorig.c? (y/n) **n**
$

cp - 파일 및 등록부를 복사한다.

인수들

-i	이미 존재하는 파일을 겹치게 하겠는가를 확정하는 통보를 내보내며 사용자의 대답에 따르는 복사를 한다.
-f	파일이름이 충돌해도 이미 존재하는 파일을 복사하는 파일로 겹쳐지게 한다.
-p	복사할 때 허락들을 보호한다.
-r	재귀적으로 복사한다.
-R	허락들이 차이나는 때만을 제외하고는 재귀적으로 복사한다.

mv

mv 지령은 파일이나 등록부를 한 장소로부터 다른 장소로 이동시킬 때 리용한다. 여러개 파일들을 동시에 이동시킬수도 있다.

다음의 실례에서는 등록부를 렬거하고 krsort.c 를 krsort.test.c 로 그 등록부안에서 이동시키며 그다음 다시 그 등록부를 렬거한다.

$ **ls -l**
total 168
```
-rwxr-xr-x   1  denise  users  34592  Oct  31  15:17  krsort
-rwxr-xr-x   1  denise  users   3234  Oct  31  15:17  krsort.c
-rwxr-xr-x   1  denise  users   2756  Oct  31  15:17  krsort.dos
-rw-r--r--   1  denise  users   9922  Oct  31  15:17  krsort.q
-rwxr-xr-x   1  denise  users   3085  Oct  31  15:17  krsortorig.c
```
$ **mv krsort.c kroort.test.c**
$ **ls -l**
total 168
```
-rwxr-xr-x   1  denise  users  34592  Oct  31  15:17  krsort
-rwxr-xr-x   1  denise  users  32756  Oct  31  15:17  krsort.dos
-rw-r--r--   1  denise  users   9922  Oct  31  15:17  krsort.q
```

```
-rwxr-xr-x   1   denise   users    3234   Oct   31   15:17   krsort.test.c
-rwxr-xr-x   1   denise   users    3085   Oct   31   15:17   krsortorig.c
$
```

만일 목적파일이 이미 존재한다면 어떤 현상이 일어 나겠는가? UNIX 는 목적파일이 이미 존재해도 겹친 상태로 이동시킨다. -i 를 리용하여 mv 지령은 이동하기전에 겹치게 하겠는가를 물어 본다. 다음의 실례는 krsort.test.c 파일을 krsortorig.c 로 이동시키는 과정을 보여 주고 있다. 사용자는 이미 존재하는 krsortorig.c 를 겹치게 하겠는가고 물어 볼 때 n 으로 대답함으로써 아무런 이동도 하지 않게 한다.

```
$ ls -l
total 168
-rwxr-xr-x   1   denise   users   34592   Oct   31   15:17   krsort
-rwxr-xr-x   1   denise   users   32756   Oct   31   15:17   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   31   15:17   krsort.q
-rwxr-xr-x   1   denise   users    3234   Oct   31   15:17   krsort.test.c
-rwxr-xr-x   1   denise   users    3085   Oct   31   15:17   krsortorig.c
$ mv -i kroort.test.c krsortorig.c
remove krsortorig.c? (y/n) n
$ ls -l
total 168
-rwxr-xr-x   1   denise   users   34592   Oct   31   15:17   krsort
-rwxr-xr-x   1   denise   users   32756   Oct   31   15:17   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   31   15:17   krsort.q
-rwxr-xr-x   1   denise   users    3234   Oct   31   15:17   krsort.test.c
-rwxr-xr-x   1   denise   users    3085   Oct   31   15:17   krsortorig.c
$
```

결국 krsortorig.c 는 초기상태대로 남아 있게 된다.

mv - 파일 및 등록부를 이동시킨다.

선택항목들

-i	이미 존재하는 파일을 겹치게 하겠는가를 확정하는 통보를 내보내며 사용자의 대답에 따르는 이동을 진행한다.
-f	파일이름이 충돌해도 이미 존재하는 파일을 이동하는 파일로 겹쳐 지게 한다.
-p	이동할 때 허락들을 보호한다.

mkdir

mkdir 지령을 리용하면 아무때나 등록부를 만들수 있다. 바로 이런 등록부창조 지령이 있기때문에 사용자들은 자기의 파일들을 여러 등록부들에 보관할수 있다. 등록부창조는 창조하려는 등록부이름을 간단히 지정하면 된다. 다음의 실례에서는 ls 지령으로 등록부의 내용을 렬거한 다음 mkdir 지령으로 default.pemissions 라는 등록부를 창조한다. 그다음에 새로운 등록부를 보기 위하여 다시 ls지령을 수행시키고 있다.

```
$ ls -l
total 2
drwxr-xr-x    2 denise users 1024  Oct 31 11:27 kraort.dir.old
$ mkdir default.permissions
$ ls -l
total 4
drwxr-xr-x    2 denise users 1024  Oct 31 11:27 kroort.dir.old
drwxr-xr-x    2 denise users       24  Oct 31 11:32 default.permissions
$
```

새로운 등록부는 denise.group 사용자의 허락을 기정값으로 창조된다.

이러한 기정값허락대신에 자기 고유의 허락을 가지고 등록부를 창조하려면 선택항목 -m 을 사용하여 mkdir 지령을 수행시키면 된다. 실례로 krsort.dir.new 등록부에 모든 사용자들이 읽기허락을 가지도록 창조하자면 다음과 같이 하면 된다.

```
$ mkdir -m -a-r- read.permissions
$ ls -l
total 6
drwxr-xr-x    2 denise users 1024  Oct 31 11:27 krsort.dir.old
drwxr-xr-x    2 denise users    24  Oct 31 11:32 default.permissions
dr--r--r--    2 denise users    24  Oct 31 11:33 read.permissions
$
```

이 mkdir 지령은 기호방식으로 허락을 리용하였다.

선택항목 -p 를 사용하면 한개 준위이상으로 규정되는 등록부를 창조할수 있다. 즉 여러개의 보조등록부를 가지는 하나의 새로운 등록부를 창조할수 있다. 실례로 level1 등록부아래에 level2, 또 그 아래에 level3 이라는 보조등록부를 가지는 level1 등록부를 창조하는 과정은 다음과 같다.

```
$ mkdir -p level1/level2/level3
$ ls -R level1
level2

level1/level2:

level3
level1/level2/level3:
$
```

mkdir - 규정된 등록부들을 창조한다

선택항목들

-m	등록부의 방식(허락)을 규정한다
-p	여러 계층의 등록부들을 창조한다

rm

rm 지령은 등록부에서 한개 혹은 그이상의 파일들을 제거하며 등록부 그자체도 제거할수 있다. 파일을 제거하자면 그 파일을 포함하는 등록부가 쓰기 및 실행허락을 둘 다 가지고 있어야 한다. 이러한 허락이 없으면 rm은 실패한다.

이미 몇개의 지령들에서 본것처럼 선택항목 -i는 매개 파일을 제거하겠는가를 확인한다. n으로 대답하면 파일은 제거되지 않으며 y로 대답하면 파일은 제거된다.

마찬가지로 선택항목 -r(-R)도 등록부의 내용을 재귀적으로 제거한 다음 그 등록부들도 제거한다. 즉 파일들을 재귀적으로 제거한 다음에 규정된 등록부들을 제거한다. 만일 파일과 등록부제거를 재귀적으로 하겠는가를 물어 보게 하려면 선택항목 -i와 선택항목 -r를 함께 사용하면 된다.

선택항목 -f를 사용하면 아무런 확인이 없이 파일과 등록부들을 제거한다.

다음의 실례는 krsort.dir.new 등록부에 대한 렬거를 하고 이 등록부안의 제거하려는 파일들을 사용자에게 물어 보면서 제거작업을 한 다음 다시 그 등록부의 내용을 렬거한다. 이 실례에서는 사용자가 모두 n으로 대답했기때문에 파일들이 하나도 제거되지 않게 된다.

```
$ ls -l krsort.dir.new
total 168
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:44  krsort
-rwxr-xr-x  1  denise  users   3234  Oct  27  18:46  krsort.c
-rwxr-xr-x  1  denise  users  32756  Oct  27  18:46  krsort.dos
```

```
-rw-r--r--    1   denise   users    9922  Oct   27   18:46   krsort.q
-rwxr-xr-x    1   denise   users    3085  Oct   27   18:46   krsortorig.c
$ rm -i krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) n
../krsort.dir.new/krsort.c: ? (y/n) n
../krsort.dir.new/krsort.dos: ? (y/n) n
../krsort.dir.new/krsort.q: ? (y/n) n
../krsort.dir.new/krsortorig.c: ? (y/n) n
$ ls -1 kroort.dir.new
total 168
-rwxr-xr-x    1   denise   users   34592  Oct   27   18:44   krsort
-rwxr-xr-x    1   denise   users    3234  Oct   27   18:46   krsort.c
-rwxr-xr-x    1   denise   users   32756  Oct   27   18:46   krsort.dos
-rw-r--r--    1   denise   users    9922  Oct   27   18:46   krsort.q
-rwxr-xr-x    1   denise   users    3085  Oct   27   18:46   krsortorig.c
$
```

등록부를 제거할 때 선택항목 **-i** 와 **-r** 를 결합하면 rm 지령과 사용자가 호상 물어보면서 작업할수 있다. 그러나 등록부안의 파일을 모두 제거하지 않으면 선택항목 **-i** 가 없을 때 그 등록부는 제거되지 않는다. 다음의 실례의 첫 부분에서는 krsort.dir.new 등록부로부터 krsort 파일이 제거되지 않는다. 따라서 이 파일이 계속 남아 있기때문에 등록부는 제거되지 않는다. 실례의 두번째 부분에서는 이 파일이 제거되기때문에 그 등록부자체도 제거된다.

```
$ rm -ir krBort.dir.new
directory krsort.dir.new: ? (y/n) y
krsort.dir.new/krsort: ? (y/n) n
krsort.dir.new/krsort.c: ? (y/n) y
krsort.dir.new/krsort.dos: ? (y/n) y
krsort.dir.new/krsort.q: ? (y/n) y
krsort.dir.new/krsortorig.c: ? (y/n) y
krsort.dir.new: ? (y/n) y
rm: directory krsort.dir.new not removed. Directory not empty
$ rm -ir krsort.dir.new
directory krsort.dir.new: ? (y/n) y
krsort.dir.new/krsort: ? (y/n) y
krsort.dir.new: ? (y/n)y
$
```

인수들

-i　이미 존재하는 파일을 제거하겠는가를 확정하는 통보를 내보내며 사용자의 대답에 따르는 제거를 진행한다

-f　파일들이 무조건 제거되게 한다

-r(-R)　등록부의 내용과 등록부 그자체를 재귀적으로 제거한다

rmdir

rmdir 지령은 한개 혹은 그이상의 등록부들을 제거한다. 제거하려는 등록부는 비여있어야 하며 그렇지 않으면 선택항목 -f 를 리용하여야 한다. 물론 하나이상의 등록부들을 제거하도록 규정할수 있다. 등록부를 제거할 때 그 등록부의 어미등록부는 쓰기 및 실행허락을 둘 다 가지고 있어야 한다.

역시 선택항목 -i 는 등록부를 제거하겠는가를 확정하며 n 으로 대답하면 제거하지 않고 y 로 대답하면 제거한다.

제거하려는 등록부들을 규정하는 순서가 중요하다. 만일 어떤 등록부를 그 등록부의 보조등록부와 함께 제거하려면 먼저 보조등록부를 규정해야 한다. 만일 제거하려는 보조등록부보다 제거하려는 어미등록부를 먼저 규정하면 그 어미등록부가 비여 있지 않기때문에 어미등록부의 제거작업은 실패한다.

선택항목 -f 를 사용하면 사용자에게 확정함이 없이 등록부들을 강제적으로 제거한다. 다음의 실례는 .dotfile 을 포함하는 krsort.dir.new 등록부를 보여 주고 있다. rmdir 로 이 등록부를 제거하려고 할 때 등록부가 비지 않았다는 통보가 나온다. .dotfile 을 제거한후에야 성과적으로 krsort.dir.new 등록부를 제거할수 있다.

```
$ ls -al ../kroort.dir.new
total 4
drwxr-xr-x   2  denise  users  1024  Oct  27  18:57  .
drwxrwxr-x   4  denise  users  1024  Oct  27  18:40  ..
-rw-r--r--   1  denise  users     0  Oct  27  18:56  dotfile
$ radir -i ../krsort.dir.new
../krsort.dir.new: ? (y/n) y
rmdir: ../krsort.dir.new: Directory not empty
$ rm ../kroort.dir.new/.dotfile
$ rmdir -i ../kroort.dir.new
../krsort.dir.new: ? (y/n) y
$
```

인수들

-i	제거하려는 등록부를 확정하고 사용자의 대답에 따르는 이동을 진행한다
-f	등록부를 무조건 제거한다
-p	등록부를 제거한후에 어미등록부가 비여 있으면 그 어미등록부도 제거한다. 이 동작은 비지 않은 등록부가 나타날 때까지 계속된다.

지령들의 사용

cd, pwd, ls, mkdir 및 cp 지령들의 사용

여기서 설명하는 지령들은 자주 사용되는 지령들이다. 이러한 지령들의 사용법에 대하여 실례를 들어 고찰해 보자.

먼저 denise의 홈등록부아래에 있는 krsort.dir.old라는 등록부계층구조를 보면 그림 3-7과 같다.

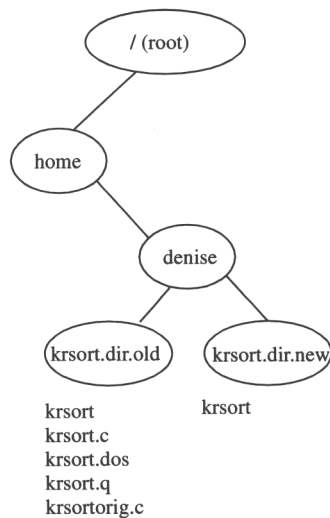



그림 3-7. /home/denise/krsort.dir.old

$ **cd /home/denise/kroort.dir.old**

$ **pwd**

/home/denise/krsort.dir.old

$ **ls -l**

total 168

```
-rwxr-xr-x   1   denise   users   34592   Oct   27   18:20   krsort
-rwxr-xr-x   1   denise   users    3234   Oct   27   17:30   krsort.c
-rwxr-xr-x   1   denise   users   32756   Oct   27   17:30   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   27   17:30   krsort.q
-rwxr-xr-x   1   denise   users    3085   Oct   27   17:30   krsortorig.c
$
```

이제 krsort.dir.new 라는 새로운 등록부를 창조하고 거기에 그림 3-8 에서와 같이 하나의 파일을 복사해 보자.

```
$ mkdir ../kroort.dir.new
$ cp krsort ../kroort.dir.new
$ ls -1 ../kroort.dir.new
total 68
-rwxr-xr-x   1   denise   users   34592   Oct   27   18:27   krsort
$
```

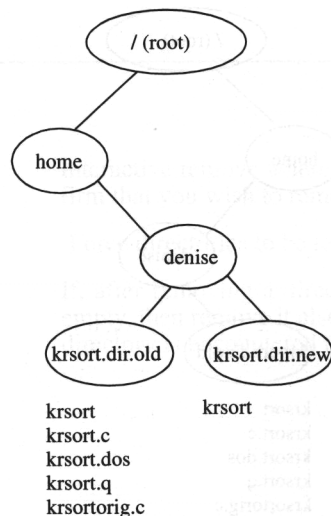



그림 3-8. /home/denise/krsort.dir.new

cp 지령에서 선택항목 -i 를 리용하여 이미 존재하는 파일이름으로 파일을 복사하려고 하면 목적파일을 겹치게 하겠는가를 물어 본다. 이때 그림 3-9 에서와 같은 방법으로 새로운 파일이름으로 복사하도록 할수 있다.

```
$ pwd
/users/denise/krsort.dir.old
$ cp -i krsort ../kroort.dir.new
```

```
overwrite ../krsort.dir.new/krsort? (y/n) n
$ cp krsort ../kroort.dir.new/kroort.new.name
$ ls -1 ../krsort.dir.new
total 136
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:27  krsort
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:29  krsort.new.name
$
```

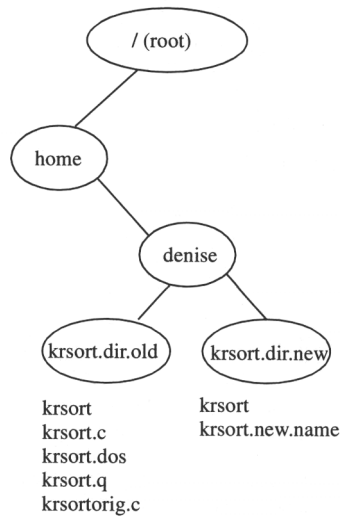



그림 3-9. /home/denise/krsort.dir.new/krsort.new.name 의 추가

또한 그림 3-10 과 같이 krsort.dir.old 에 있는 모든 파일들을 krsort.dir.new 에 복사하기 위하여서는 통용기호를 가지고 지령 cp 를 리용한다.

```
$ cp * ../kroort.dir.new
$ ls -l ../krsort.dir.new
total 236
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:30  krsort
-rwxr-xr-x  1  denise  users   3234  Oct  27  18:30  krsort.c
-rwxr-xr-x  1  denise  users  32756  Oct  27  18:30  krsort.dos
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:29  krsort.new.name
-rw-r--r--  1  denise  users   9922  Oct  27  18:30  krsort.q
-rwxr-xr-x  1  denise  users   3085  Oct  27  18:30  krsortorig.c
$
```

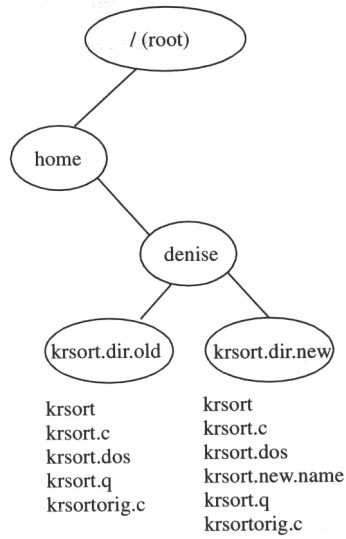



그림 3-10. /home/denise/krsort.dir.new 에 복사된 모든 파일들

mv 지령의 사용

그림 3-11 과 같이 krsort.dir.new 등록부가 비여 있는 시점에서 고찰하자.

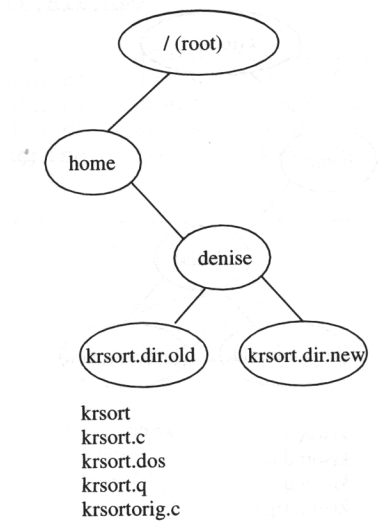



그림 3-11. 빈 /home/denise/krsort.dir.new 등록부

그림 3-12 와 같이 krsort 파일을 krsort.dir.new 등록부로 이동시킬수 있다.

```
$ mv krsort ../krsort.dir.new
$ ls -l ../krsort.dir.new
total 68
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:20  krsort
$
```

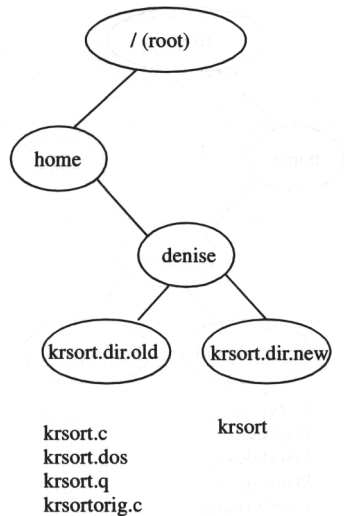



그림 3-12. /home/denise에 이동된 krsort

만일 krsort를 선택항목 -i를 가지고 krsort.dir.new 등록부에 이동시키려고 한다면 krsort파일이 겹치게 되는 경우에는 다음의 통보를 받게 된다.

```
$ mv -i krsort ../kroort.dir.now/kroort
remove ../krsort.dir.new/krsort? (y/n) n
$
```

여기서는 선택항목 -i를 가지고 mv를 리용하였기때문에 파일을 겹치게 이동시키겠는가를 확정하는 통보를 받게 된다. 이때 n으로 대답하면 파일은 겹치지 않게 된다.

mv 지령에서 통용기호를 사용하여 krsort.dir.old 등록부로부터 krsort.dir.new 등록부에로 모든 파일을 이동시킬수 있다. 선택항목 -i가 없으면 그림 3-13에서와 같이 krsort.dir.new 등록부에 있는 임의의 파일은 동일한 이름을 가진 파일로 겹쳐 지게 된다.

```
$ mv * ../kroort.dir.new
$ ls -l ../kroort.dir.new
total 168
-rwxr-xr-x  1  denise  users  34592  Oct  27  18:44  krsort
-rwxr-xr-x  1  denise  users   3234  Oct  27  18:46  krsort.c
```

```
-rwxr-xr-x   1   denise   users   32756   Oct   27   18:46   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   27   18:46   krsort.q
-rwxr-xr-x   1   denise   users    3085   Oct   27   18:46   krsortorig.c
$
```

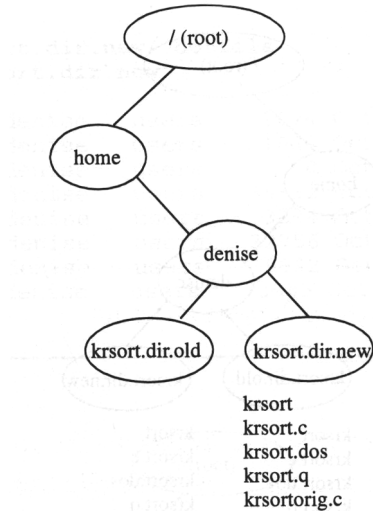



그림 3-13. /home/denise/krsort.dir.new에 이동된 모든 파일들

rm 및 rmdir 지령의 사용

UNIX 체계에서 제일 근심되는 지령이 rm 지령이다. 왜냐하면 rm 지령은 선택항목 -i만 없으면 아무런 확인도 없이 언제든지 무엇이나 제거할수 있기때문이다.

그러나 rm 지령은 사용자들을 도와 주는 측면이 더 많다. 체계관리자가 아니라 평범한 사용자인 경우에 자기의 모든 파일과 등록부들을 제거하는 허락을 가지기때문에 단순히 선택항목 -i를 가지고 rm을 사용하면 대화식으로 유연하게 작업할수 있다. 그림 3-14에서와 같이 krsort.dir.new와 krsort.dir.old가 동일한 등록부라고 가정하자.

krsort.dir.new에서 파일들을 대화식으로 제거하기 위해서는 다음과 같이 한다.

```
$ rm -i ../krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) n
../krsort.dir.new/krsort.c: ? (y/n) n
../krsort.dir.new/krsort.dos: ? (y/n) n
../krsort.dir.new/krsort.q: ? (y/n) n
../krsort.dir.new/krsortorig.c: ? (y/n) n
$ ls -l ../krsort.dir.new
total 168
-rwxr-xr-x   1   denise   users   34592   Oct   27   18:44   krsort
```

```
-rwxr-xr-x   1   denise   users    3234   Oct   27   18:46   krsort.c
-rwxr-xr-x   1   denise   users   32756   Oct   27   18:46   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   27   18:46   krsort.q
-rwxr-xr-x   1   denise   users    3085   Oct   27   18:46   krsortorig.c
$
```

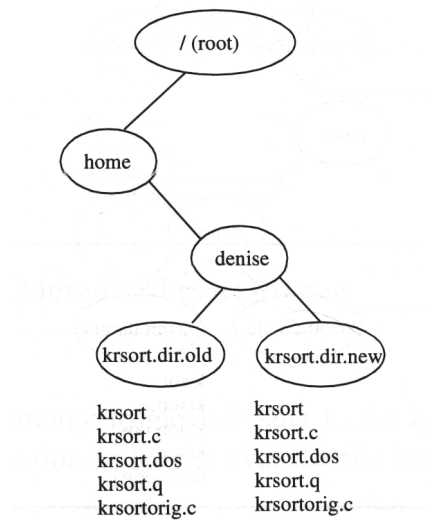



그림 3-14. 동일한 /home/denise/krsort.dir.old 와 /home/denise/krsort.dir.new

우의 실례에서는 제거하려는 파일들에 대하여 모두 n 으로 대답했기때문에 krsort.dir.new 등록부로부터 아무런 파일도 제거되지 않았다.

더 나아가서 "."으로 시작되는 하나의 파일을 krsort.dir.new 에 추가해 보자. touch 지령은 그림 3-15 에서와 같이 내용이 없이 파일을 창조만 한다(touch 지령은 이미 존재하는 파일의 시간속성을 갱신하는데도 리용할수 있다.).

```
$ touch ../krsort.dir.now/.dotfile
$ ls -al ../krsort.dir.new
total 172
drwxr-xr-x   2   denise   users    1024   Oct   27   18:54   .
drwxrwxr-x   4   denise   users    1024   Oct   27   18:40   ..
-rw-r--r--   1   denise   users       0   Oct   27   18:56   dotfile
-rwxr-xr-x   1   denise   users   34592   Oct   27   18:44   krsort
-rwxr-xr-x   1   denise   users    3234   Oct   27   18:46   krsort.c
-rwxr-xr-x   1   denise   users   32756   Oct   27   18:46   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   27   18:46   krsort.q
-rwxr-xr-x   1   denise   users    3085   Oct   27   18:46   krsortorig.c
$
```

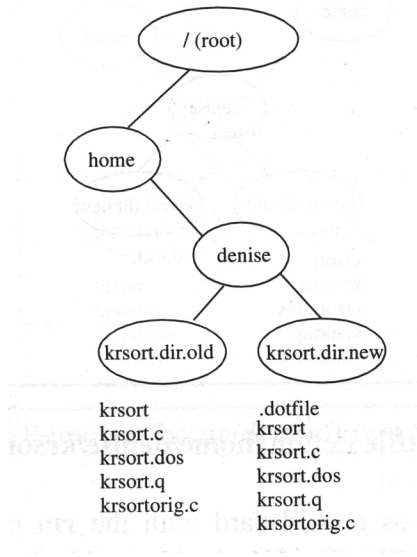



그림 3-15. .dotfile 이 있는 /home/denise/krsort.dir.new

다음으로 그림 3-16 에서와 같이 rm 지령으로 파일들을 제거하려면 다음과 같이 한다.

```
$ rm -i ../krsort.dir.new/*
../krsort.dir.new/krsort: ? (y/n) y
../krsort.dir.new/krsort.c: ? (y/n) y
../krsort.dir.new/krsort.dos: ? (y/n) y
../krsort.dir.new/krsort.q: ? (y/n) y
../krsort.dir.new/krsortorig.c: ? (y/n) y
$ ls -al ../kroort.dir.new
total 4
drwxr-xr-x   2  denise  users  1024  Oct  27  18:57  .
drwxrwxr-x   4  denise  users  1024  Oct  27  18:40  ..
-rw-r--r--   1  denise  users     0  Oct  27  18:56  dotfile
```

우의 실례에서 rm 지령에서 사용한 통용기호 "*"는 .dotfile 을 제거하지 못한다. 이 파일이 있으면 rmdir 지령은 krsort.dir.new 등록부를 제거할수 없다. rmdir 가 krsort.dir.new 를 성과적으로 제거하려면 먼저 이 파일이 제거되여야 한다.

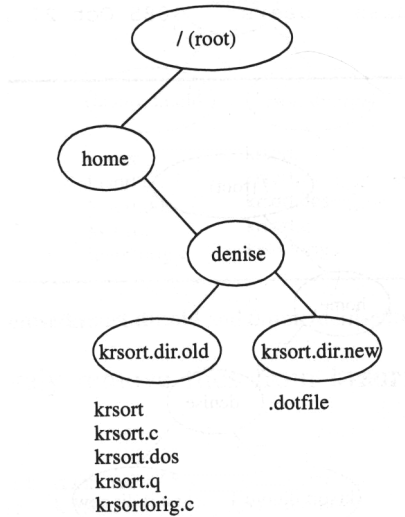



그림 3-16. .dotfile 만이 남아 있는 /home/denise/krsort.dir.new

```
$ rmdir -i ../kroort.dir.new
../krsort.dir.new: ? (y/n) y
rmdir: ../krsort.dir.new: Directory not empty
$ rm ../krsort.dir.new/.dotfile
$ rmdir -i ../kroort.dir.new
../krsort.dir.new: ? (y/n) y
$
```

rmdir 는 그림 3-17 에서와 같이 .dotfile 이 제거되였기때문에 krsort.dir.new 를 성과적으로 제거하였다.

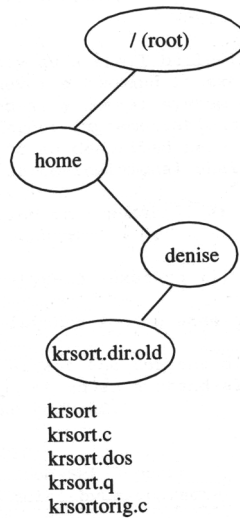



그림 3-17. rmdir 는 /home/denise/krsort.dir.new 를 제거한다.

지 령 소 개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 약간 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들이 가장 우수한 기준으로 된다.

cd

cd - 작업등록부를 변경시킨다

---

cd(1)

이름

cd - 작업등록부를 변경시킨다

형식

cd [directory]

해설

등록부가 규정되지 않으면 쉘파라메터 HOME 의 값이 새로운 작업등록부로 리용된다. 등록부를 /, ., .., 으로 시작하는 완전경로로 규정하면 그 등록부는 새로운 작업등록부로 된다. 이 두 경우가 아닌 때에 cd 는 쉘파라메터 CDPATH 로 규정한 경로에 대하여 상대적인 목적등록부를 찾는다. CDPATH 는 쉘파라메터 PATH 와 문법적으로 동일하며 의미적으로 류사하다. cd 는 등록부에 대한 실행(탐색)허락을 가져야 한다.

cd 는 쉘의 체계지령으로서만 될수 있다. 왜냐하면 지령이 실행될 때마다 새로운 과제가 창조되면서 cd 가 보통의 체계지령으로 씌여 지고 처리되는 경우에는 그 지령을 무시해 버리기때문이다. 더우기 각이한 쉘들이 cd 의 각이한 실현들을 체계봉사프로그람으로서 제공하고 있다. 여기서 서술하는 cd 의 기능들은 모든 쉘들에서 제공되지 않을수도 있다(개별적인 쉘지령들을 참고).

만일 cd 가 다음과 같은 부분쉘 혹은 개별적인 봉사프로그람실행환경에서 호출된다면 cd 는 호출측환경의 현재등록부에는 영향을 주지 않는다.

```
find .-type d -exec cd {}; -exec foo {};
```

(이것은 접근가능한 등록부들에 foo 를 입장시킨다.)

외부적영향

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

파일체계의 임의의 위치로부터 HOME 등록부에로 현재작업등록부를 변경시킨다.

```
cd
```

현재등록부에 남아 있는 foo 에로 현재작업등록부를 새롭게 변경시킨다.

```
cd foo
```

혹은

```
cd ./foo
```

현재등록부의 어미등록부에 남아 있는 foobar 등록부로 변경시킨다.

```
cd ../foobar
```

절대경로 /usr/local/lib/work.files 로 변경시킨다.

```
cd /usr/local/lib/work.files
```

홈등록부에 대한 상대적인 등록부 proj1/schedule/staffing/proposals 로 변경시킨다.

```
cd $HOME/proj1/schedule/staffing/proposals
```

변수들

다음의 환경변수들은 cd 의 실행을 변경시킨다.

HOME 홈등록부이름, 등록부인수가 규정되지 않았을 때 사용된다.

CDPATH 등록부를 참조하는 두 점으로 구분된 경로이름들의 목록이다.

만일 등록부가 사선(/)으로 시작하지 않거나 첫 성분이 점(.) 또는 두점(..)이 아니면 cd 는 CDPATH 변수에 들어 있는 매개 등록부에 대한 상대적인 등록부를 탐색한다. 새로운 작업등록부는 이 목록에서 처음으로 일치하는 등록부로 설정된다. 등록부경로이름위치의 빈 기호렬은 현재의 등록부를 나타낸다. CDPATH 가 설정되지 않았으면 이것은 빈 기호렬로 취급된다.

귀환값

다음의 값들중의 하나로 귀환한다.

0	등록부가 성과적으로 변경되였다.
>0	오유가 발생하였다. 작업등록부는 변경되지 않은채로 남아 있다.

관련항목

csh(1), pwd(1), ksh(1), sh-posix(1), sh(1), chdir(2)

표준일치

cd: SVID2, SVID3, XPG2, XPG4, POSIX.2

chmod

chmod - 허락을 변경시킨다.

---

chmod(1)

이름

chmod - 파일의 방식 즉 접근허락을 변경시킨다.

형식

/usr/bin/chmod [-A] [-R] symbolic_mode_list file …

절대형식:

/usr/bin/chmod [-A] [-R] numeric_mode file …

해설

chmod 지령은 기호방식목록이나 수값방식의 값에 따라 하나 혹은 그이상의 파일들의 허락을 변경시킨다. 사용자는 ls -l 지령으로 파일에 대한 현재의 허락을 현시할수 있다(ls(1)을 참고).

기호방식목록

기호방식목록은 다음의 형식에서와 같이 반점으로 구분된 연산자들의 목록이다. 이 형식에서 공백은 허용되지 않는다.

[who] op [permission] [, …]

변수마당은 다음의 값들을 가진다.

who 다음의 문자들중에서 하나 혹은 그이상을 가진다.

- u 사용자(소유자)의 허락을 수정한다.
- g 그룹의 허락을 수정한다.
- o 기타 사용자들의 허락을 수정한다.
- a 모든 사용자들의 허락을 수정한다(a 는 ugo 와 동등하다).

op 다음기호들중에서 하나를 취한다.

- \+ 주어 진 파일에 who 로 규정된 방식비트의 허락들을 추가한다.
- \- 주어 진 파일로부터 who 로 규정된 방식비트의 허락들을 제거한다.
- = who 로 규정된 방식비트의 허락들을 교체한다.

permission

다음의 문자들중에서 하나 혹은 그이상을 가진다.

- r who 에 대한 읽기허락을 추가/제거한다.
- w who 에 대한 쓰기허락을 추가/제거한다.
- x who 에 대한 실행(등록부탐색)허락을 추가/제거한다.
- s who 에 대한 실행허락의 사용자 ID 설정이나 그룹 ID 설정을 추가/제거한다. who 에서 u 혹은 g 가 설정되였을 때만 가능하다.

t　　파일실행허락(보호비트)에 대한 본문 및 화상보관을 추가/제거한다(chmod(2)를 참고).

X　　실행/탐색허락을 다음과 같은 조건에서 추가 또는 제거한다.

- 등록부인 경우 who에 대한 탐색허락을 추가 또는 제거한다(x와 동일하다.).
- 등록부가 아니고 현재 파일허락이 적어도 하나의 사용자, 그룹, 기타에 대한 실행허락을 포함한다면(ls -l이 x 또는 s를 현시한다면) who에 대한 실행허락을 추가 또는 제거한다.
- 등록부가 아니고 현재 파일의 실행허락이 그 누구에게도 설정되여 있지 않다면 실행허락을 전혀 변경시키지 않는다.

또는 다음의 문자들중의 하나만을 가진다.

u　　현재 소유자허락을 who에 복사한다.

g　　현재 그룹허락을 who에 복사한다.

o　　현재 기타허락을 who에 복사한다.

연산자들은 규정된 순서로 수행되며 동일한 지령행에서 이미 규정된 연산자들도 중복될수 있다.
who가 생략되면 현재파일방식창조마스크가 변경을 허락하는 경우에 모든 사용자들의 r, w, x, X 허락들이 변경된다(umask(1)을 참고). s 및 t 허락들은 who안에서 규정된듯이 변화된다.
허락들을 생략하는것은 모든 허락들을 제거하기 위하여 =를 리용할 때에만 효과적이다.

수값방식(절대적방식)

절대적인 허락들은 수값방식으로 규정할수 있다. 수자적허락들은 다음과 같은 방식비트들의 론리적합(OR)으로 규정되는 8진수들이다.

사용자방식비트:

4000 (= u = s)　　파일실행에 대한 사용자ID설정(파일만 가능)

2000 (= g = s)　　파일실행에 대한 그룹ID설정

1000 (= u = t)　　본문 및 화상보관설정(chmod(2)를 참고)

허락방식비트:

0400	(= u = r)	소유자의 읽기
0200	(= u = w)	소유자의 쓰기
0100	(= u = x)	소유자의 실행(등록부탐색)
0040	(= g = r)	그룹읽기
0020	(= g = w)	그룹쓰기
0010	(= g = x)	그룹실행(탐색)
0004	(= o = r)	기타 읽기
0002	(= o = w)	기타 쓰기
0001	(= o = x)	기타 실행(탐색)

선택항목들

-A 파일과 관련되는 선택적인 접근조종목록(ACL)을 예견한다. 기정값으로는 IEEE 표준 POSIX 1003.1-1988 과 일치되며 선택적인 ACL 구체례들은 제거된다(접근조종목록에 대해서는 acl(5)를 참고).

-R 파일방식비트들을 재귀적으로 변경시킨다. 등록부로 되는 매개 파일인수에 대하여 chmod 는 그 파일계층에서 아래에 있는 모든 파일 및 보조등록부들의 파일접근비트를 변경시킨다.

파일의 소유자 혹은 특권을 가진 사용자만이 그것의 방식을 변경시킬수 있다.

또한 특권을 가진 사용자만이 파일의 보호비트를 설정할수 있다.

확장비트로 그룹 ID 설정을 규정하자면 파일그룹이 현재그룹 ID 에 대응되여야 한다.

만일 chmod 가 기호적련결에 리용된다면 련결에 의하여 참조되는 파일의 방식이 변경된다.

외부적영향

환경변수들

LC_MESSAGES 는 통보가 현시되는 언어를 결정한다. 만일 이 변수가 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 무효한 설정을 포함하면 chmod 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

귀환값

다음의 값들중의 하나로 귀환한다.

0	등록부가 성공적으로 완성
>0	오유가 발생

실례

기타에 대한 쓰기허락을 부정한다.

```
chmod o-w file
```

모든 사용자들의 실행을 허용한다.

```
chmod a+x file
```

모든 사용자들에게 읽기 및 실행허락을 주고 사용자ID설정비트를 규정한다.

```
chmod a=rx, u+s file
```

파일소유자에게 읽기 및 쓰기 허락을 주고 모든 사용자에게 읽기허락을 준다.

```
chmod u=rw, go=r file
```

또는

```
chmod 644 file(절대적인 형식)
```

사용자와 그룹에게는 등록부계층나무의 모든 파일들을 읽도록 하며 모든 사용자들에게는 실행할수 있게 한다.

```
chmod -R ug+r, o-r, a+X pathname
```

만일 umask의 현재값이 020(umask -S는 u=rwx, g=rx, o=rwx; 을 현시하며 그룹의 쓰기허락은 변경시키지 않는다.)이고 파일 mytest의 현재 허락이 444(a=r)이며 ls -l이 -r--r--r--을 현시한다면 다음의 지령은 646허락(uo=rw, g=r)을 설정한다. 결국 ls -l은 -rw-r--rw-를 현시한다.

```
chmod +w mytest
```

만일 umask의 현재값이 020(umask -S는 u=rwx, g=rx, o=rwx; 을 현시하며 그룹의 쓰기허락은 변경시키지 않는다.)이고 파일 mytest의 현재허락이 666(a=rw)이며 그리고 ls -l이 -rw-rw-rw-를 현시한다면 다음의 지령은 464허락(uo=r, g=rw)을 설정한다. 결국 ls -l은 -r--rw-r--를 현시한다.

```
chmod -w mytest
```

종속성

ACL 을 지원하지 않는 파일체계에서 선택항목 -A 를 리용하면 chmod 는 실패한다.

저자

chmod 는 AT&T 와 HP 에서 개발하였다.

관련항목

chacl(1), ls(1), umask(1), chmod(2), acl(5)

표준일치

chmod: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

cp

cp - 파일 및 등록부를 복사한다

---

cp(1)

이름

cp - 파일 및 보조등록부나무를 복사한다

형식

cp [-f | -i] [-p] [-e extarg] file1 new_file

cp [-f | -i] [-p] [-e extarg] [file2 …] dest_directory

cp [-f|-i] [-p] [-R|-r] [-e extarg] directory1 [directory2, …] dest_directory

해설

cp 는 다음과 같이 복사를 진행한다.

- file1 을 새롭게 혹은 이미 존재하는 new_file 에 복사한다.
- file1 을 이미 존재하는 dest_directory 에 복사한다.
- file1, file2, …을 이미 존재하는 dest_directory 에 복사한다.
- 보조등록부나무 directory1 을 새롭게 혹은 이미 존재하는 dest_directory 에 복사한다.
- 여러개의 보조등록부나무 directory1, directory2, …을 새롭게 혹은 이미 존재하는 dest_directory 에 복사한다.

cp 는 file1 과 new_file 이 같으면 실패한다. 목표가 등록부라면 한개이상의 파일이 그 등록부안에 복사된다. 두개이상의 파일을 복사하려면 목표가 반드시 등록부여야 한다. 하나의 파일을 new_file 에 복사할 때 new_file 이 이미 존재하면 존재하는 파일의 원래 내용은 없어 진다.

목적등록부나 이미 존재하는 파일에 대한 접근허락이 쓰기를 금지한다면 cp 는 실패하고 "cannot create file"이라는 통보를 내보낸다. 한개이상의 보조등록부나무를 다른 등록부에 복사하려면 선택항목 -r 가 필요하다. 선택항목 -r 는 하나의 파일을 다른 파일로 복사하거나 파일들을 하나의 등록부에 복사하는 경우에는 무시된다.

만일 new_file 이 다른 련결을 가진 이미 존재하는 파일에 대한 련결이라면 cp 는 이미 존재하는 파일을 겹치게 하며 모든 련결을 그대로 남겨 둔다. 파일을 이미 존재하는 파일에로 복사할 때 cp 는 이미 존재하는 접근허락비트들과 소유자 및 그룹을 변경시키지 않는다.

파일들을 이미 존재하지 않는 등록부나 파일에로 복사할 때 cp 는 file1 의 허락비트를 가지고 새로운 파일을 창조하며 선택항목 -p 가 규정되지 않으면 새로운 파일의 허락비트렬은 사용자의 파일창조마스크에 의하여 S_IRWXU 와의 배타적합으로 수정된다. 새로운 파일의 사용자속성은 복사하는 파일의 사용자속성과 같아 진다. new_file(이미 존재하지 않았다면)의 최종수정시간(및 최종접근시간)과 원천 file1 의 최종접근시간은 복사한 시간으로 설정된다.

선택항목들

-i 대화식복사를 한다. cp 가 표준적인 오유들에 대하여 통보를 내보내고 이미 존재하는 파일을 겹치게 복사하겠는가를 물어 본다. 복사가 허용되면 복사된다. -i 와 -f 가 둘 다 규정되면 -i 는 무시된다.

-f 복사하기전에 아무런 확인이 없이 이미 존재하는 목적경로이름들을 제거한다. 이 선택항목은 복사에 의해서 창조되는 새로운 파일의 이름 및 등록부위치가 이미 존재하는 이름 및 위치와 충돌하는 경우에 그것을 파괴하거나 교체한다.

-p 허락을 예약한다. 즉 수정시간, 접근시간, 파일방식, 사용자 ID, 그룹 ID 등을 cp 가 허락으로 제공하도록 한다.

-r 재귀적인 부분나무복사를 한다. 이 선택항목은 cp 가 매개 원천등록부로 시작하는 부분나무를 목적등록부로 복사하도록

한다. 만일 dest_directory 가 이미 존재하면 cp 는 그안에 file1 과 동일한 이름을 가진 등록부를 창조하고 file1 에서 시작되는 부분나무를 dest_directory/file1 에 복사한다. dest_directory 가 존재하지 않으면 cp 는 그것을 창조하고 file1 로 시작되는 부분나무를 dest_directory 등록부에 복사한다. cp -r 는 부분나무를 결합하지 않는다.

보통 문자장치, 블로크장치, 망파일, 파이프, 기호적련결, 소케트와 같은 표준적인 파일들과 등록부들이 복사된다. 그러나 이때 사용자가 그 파일들에 대한 접근허락을 가지고 있어야 한다. 그렇지 않으면 파일이 창조될수 없다는 경고가 나오며 파일은 복사되지 않는다. dest_directory 는 directory1 에 존재하지 말아야 하며 또한 directory1 은 순환적인 등록부구조를 가지지 말아야 한다. 그렇지 않으면 이 두 경우에 cp 는 자료를 무한정 복사하려고 하게 된다.

-R 재귀적인 부분나무복사를 한다. 이 선택항목은 -r 와 같으나 -R 에 의하여 복사되는 등록부들이 소유자에 대한 읽기, 쓰기, 탐색허락을 가지고 창조된다는것만이 차이난다. 기타 사용자 및 그룹에 대한 허락들은 변경되지 않는다.

-R 및 -r 선택항목을 리용하면 cp 는 표준적인 파일 및 등록부들외에 FIFO, 문자 및 블로크장치파일들, 기호적련결 등도 복사한다. 그러나 **상급사용자**(Super User)들만이 장치파일을 복사할수 있다. 그밖의 사용자들이 하면 오유가 발생한다. 기호적련결들도 목적위치에 복사된다.

경고: 장치고유파일들을 가지는 등록부나무를 복사할 때 -r 를 리용하여야 한다. 그밖의 경우에는 장치고유파일로부터 무한한 자료량이 읽어 지며 그것은 파일체계의 큰 공간을 차지하면서 목적등록부안에 있는 고유파일에 중복된다.

-e extarg

복사되는 파일들의 확장속성들을 조종하게 한다. extarg 는 다음값들중의 하나를 취한다.

warn	확장속성들이 복사될수 없을 때에 경고통보를 내보낸다.
ignore	확장속성들을 복사하지 않는다.
force	확장속성들을 복사할수 없으면 파일복사는 실패한다.

목적의 파일체계가 확장속성들을 지원하지 않거나 초기의것과 다른 블로크크기를 가진다면 확장속성들은 복사될수 없다. -e가 규정되지 않으면 기정값은 warn이다.

접근조종목록(ACL)

new_file이 새로운 파일이거나 새로운 파일이 dest_directory에 창조된다면 원래의 file1, file2, … 등에 대한 접근조종목록이 계승된다(acl(5)를 참고).

외부적영향

환경변수들

LC_CTYPE는 본문에 대한 단일/다중바이트문자들의 해석을 결정한다.

LANG 및 LC_CTYPE는 yes/no질문의 y와 동등한 국부언어를 결정한다.
LANG은 통보를 현시하는 언어를 결정한다.
만일 LC_CTYPE가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG의 값이 기정값으로 리용된다. LANG이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 cp는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

다음의 지령은 sourcedir 등록부와 그 등록부의 내용을 새로운 위치에 있는 targetdir 등록부에로 복사한다. 여기서 cp는 새로운 등록부를 창조하기때문에 목적등록부 targetdir는 이미 존재하지 말아야 한다.

```
cp -r sourcedir targetdir && rm -rf sourcedir
```

-r는 sourcedir 등록부안에 있는 부분나무(파일 및 보조등록부들)를 targetdir에 복사한다. &&는 조건적인 작용을 규정한다. 즉 &&의 왼변의 연산이 성공하면 오른변이 수행된다(낡은 등록부를 제거한다.). &&의 왼변이 성공하지 못하면 낡은 등록부는 제거되지 않는다. 이 실례는 다음의 실례와 동등하다.

```
mv sourcedir targetdir
```

현재 등록부안의 모든 파일과 보조등록부나무들을 복사하려면 다음과 같이 한다.

```
cp -r * targetdir
```

sourcedir 안의 모든 파일과 보조등록부나무들을 복사하려면 다음과 같이 한다.

```
cp -r sourcedir/* targetdir
```

sourcedir 와 targetdir 앞에 등록부경로이름을 각각 줄수 있다.
길이가 령인 파일을 창조하려면 다음것들중의 하나로 할수 있다.

```
cat /dev/rul >file
cp /dev/rul file
touch file
```

종속성

NFS
망파일의 접근조종목록들은 stat()에 의한 st_mode 로 귀환될 때 집약되지만 새로운 파일에 복사되지는 않는다. 이런 파일들에 대하여 mv 혹은 ln 지령을 리용할 때 파일의 겹쳐쓰기를 확인해 보면 방식값뒤에 +는 출력되지 않는다.

저자

cp 는 AT&T 와 캘리포니아종합대학, 버클리, HP 에서 개발하였다.

관련항목

cpio(1), ln(2), mv(1), rm(1), link(1M), lstat(2), readlink(2), stat(2), symlink(4), acl(5)

표준일치

cp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

mkdir

mkdir - 등록부를 창조한다.

---

mkdir(1)

이름

mkdir - 등록부를 창조한다.

형식

mkdir [-p] [-m mode] dirname…

해설

mkdir 는 규정된 등록부들을 07777 방식(-m mode 가 규정되지 않았다면 가능한 범위에서 umask 에 의하여 변경된다.)으로 창조한다(umask(1)을 참고).
표준등록부들(등록부 그자체)과 그것의 어미등록부들은 자동적으로 창조된다.
dirname 이 이미 존재하면 mkdir 는 진단통보를 내보내고 탈퇴하며 그 등록부는 변경되지 않는다.

선택항목들

-m mode

규정된 등록부를 창조한후에 파일허락들은 mode 로 설정된다. mode 는 chmod 로 정의된 기호방식의 기호렬이다(chmod(1)을 참고). umask(1)은 -m 보다 높은 우선권을 가진다.

-p

필요할 때마다 중간등록부들을 창조한다. 그밖의 경우에는 dirname 의 완전경로가 이미 존재해야 한다. dirname 인수의 완전경로에 있는 매개 등록부이름에 대하여 그것이 이미 존재하지 않으면 규정된 등록부는 현재의 umask 설정을 리용하여 창조된다. 그러나 이때 umask 의 설정에는 무관계하게 mkdir 가 보다 낮은 등록부를 창조할수 있다는것을 담보하도록 chmod u+wx 가 매 성분에 대하여 수행된다. dirname 인수의 완전경로에 있는 매개 등록부의 이름이 이미 존재한다면 오유없이 그 이름은 무시된다. 중간경로성분들이 존재하지만 그 성분들에 대한 쓰기나 탐색허락을 가지지 않으면 mkdir 는 오유로 실패한다. dirname 인수자체가 이미 존재하는 등록부이면 mkdir 는 오유로 실패한다. -m 이 리용되면 dirname 으로 규정된 등록부는 규정된 허락으로 창조된다. 다만 LINK_MAX 보조등록부들만이 창조될수 있다(limits(5)를 참고).

외부적영향

환경변수들

LANG 은 설정되지 않았거나 비여 있는 국제화변수들에 대한 기정값을 제공한다. 만일 LANG 이 결정되지 않았거나 빈기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 mkdir 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모

든 값을 무시한다.
LC_CTYPE 는 본문의 단일/다중바이트문자들, 인쇄가능한 문자들의 분류, 정규식에서 문자클라스식들에 대한 해석을 결정한다.

LC_MESSAGES 는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용과 표준적인 출력장치에 씌여 지는 비형식적인 통보를 위한 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임보장
단일바이트 및 다중바이트기호코드모임이 보장된다.

진단
mkdir 는 모든 등록부들이 성과적으로 만들어 지면 코드 0 을 귀환하며 그렇지 못하면 진단통보를 내보내고 령이 아닌 값을 귀환한다.

실례
현재등록부에서 이미 존재하는 등록부 raw 의 아래에 gem 등록부를 창조하려면 다음과 같이 한다.

mkdir raw/gem

현재등록부아래에 raw/gem/diamond 등록부를 창조하고 그 등록부에 모든 사용자들이 읽기만 하는 허락을 설정하려면 다음과 같이 한다.

mkdir -p -m "a=r" raw/gem/diamond

이 지령은 다음의 지령과 동등하다(chmod(1)을 참고).

mkdir -p -m 444 raw/gem/diamond

만일 raw 혹은 raw, gem 등록부가 이미 존재한다면 그 등록부들은 규정된 경로에서 창조되지 않는다.

관련항목
rm(1), sh(1), umask(1)

표준일치
mkdir: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

mv

mv - 파일 및 등록부들을 이동시키거나 이름을 바꾼다.

mv(1)

이름

mv - 파일 및 등록부들을 이동시키거나 이름을 바꾼다.

형식

mv [-f|-i] [-e extarg] file1 new_file
mv [-f|-i] [-e extarg] file1 [file2 …] dest_directory
mv [-f|-i] [-e extarg] directory1 [directory2 …] dest_directory

해설

mv는 다음과 같이 동작을 수행한다.

- 한개의 파일(file 1)을 새롭게 혹은 이미 존재하는 파일(new_file)에로 이동시킨다.
- 한개이상의 파일(file 1, [file 2, …])을 이미 존재하는 등록부(dest_directory)에로 이동시킨다.
- 한개이상의 보조등록부나무(directory 1, [directory 2, …])를 새롭게 혹은 이미 존재하는 등록부(dest_directory)에로 이동시킨다.

file1 을 new_file 로 이동시키는것은 등록부안에서 파일이름을 변경시키거나 그 파일체계 혹은 다른 파일체계에서 파일의 위치를 다시 규정하는것과 같다. 목적지가 등록부일 때 한개이상의 파일들이 그 등록부에 이동된다. 두개이상의 파일이 이동된다면 목적지는 반드시 등록부여야 한다. 한개의 파일을 new_file 로 이동시킬 때 new_file이 이미 존재하면 그 파일의 내용은 파괴된다.

dest_directory 혹은 이미 존재하는 new_file에 대한 접근허락이 쓰기를 금지한다면 mv 는 그 파일을 겹쳐 쓰겠는가고 물어 본다. 이때 표준입력장치로부터의 yes/no 대답에 따라 그 방식을 결정한다(chmod(2) 및 아래의 접근조종항목을 참고).

만일 file1 이 파일이고 new_file 이 다른 련결을 가진 또 다른 파일이라면 다른 련결들은 그대로 남아 있고 new_file 은 새 파일로 된다. 만일 file1 이 련결을 가진 파일이거나 어떤 파일에 대한 련결이라면 그 파일 혹은 련결은 그대로

남아 있고 그 이름이 new_file 로 변한다. 이때 new_file 은 mv 지령에서 리용한 등록부경로이름에 따라 file1 이 남아 있는 등록부안에 있을수도 있고 없을수도 있다. 파일들에 대한 최종접근 및 수정시간은 변하지 않고 그대로 리용된다.

선택항목들

-f
: 허락에 대한 대화가 없이 mv 지령을 수행한다. 이것은 표준장치가 말단이 아닐 때 진행된다.

-i
: mv 는 이미 존재하는 파일에 겹쳐 이동하기전에 대화식으로 표준출력장치에서 확인을 한다. 표준입력장치로부터의 대답이 긍정적이면 파일은 이동에 대한 허락이 허락된것으로 보고 이동된다.

-e extarg
: 이동해야 할 파일들의 확장속성들을 조종하게 한다. extarg 는 다음 값들중의 하나를 취한다.

  warn
  : 확장속성들이 보존될수 없고 파일은 이동될수 있을 때 경고통보를 내보낸다.

  ignore
  : 확장속성들을 보존하지 않는다.

  force
  : 확장속성들을 보존할수 없으면 파일을 이동시키지 않는다.

  만일 여러개의 원천파일들이 한개의 목적등록부로 규정되면 mv 는 확장속성들을 가지지 않는 파일들을 이동하거나 보호될수 있는 확장속성들을 가지는 파일들을 이동시킨다. 결국 mv 는 확장속성들을 보호할수 없는 파일들은 이동시키지 않는다.

: 확장속성들은 파일들이 확장속성을 보장하지 않는 파일체계로 이동될 때에는 보호될수 없다. 또한 초기의것과 다른 블로크크기를 가지는 파일체계로 이동될 때에도 보호될수 없다. -e 가 규정되지 않으면 기정값은 warn 이다.

접근조종목록(ACL)

ACL 구체례들이 선택적으로 new_file 과 관련되여 있다면 mv 는 파일에 대한 겹친 허락을 확인하는 접근방식뒤에 +를 현시한다.

new_file 이 새로운 파일이면 그것은 file1 의 접근조종목록을 계승하며 두 파일들의 소유자관계에서의 차이점이 반영되게 한다(acl(5)를 참고).

외부적영향

환경변수들

LC_CTYPE 는 본문의 단일/다중바이트문자들의 해석을 결정한다.

LANG 및 LC_CTYPE 는 yes/no 질문에서의 y 와 동등한 국부적언어를 결정한다.
LANG 은 통보가 현시되는 언어를 결정한다.
만일 LC_CTYPE 가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 mv 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

현재등록부에서 파일의 이름을 바꾸려면 다음과 같이 한다.

```
mv  old_filename  new_filename
```

현재등록부에서 등록부의 이름을 바꾸려면 다음과 같이 한다.

```
mv  old_dirname  new_dirname
```

현재등록부에서 인쇄되지 않는 조종문자 혹은 -, *와 같은 쉘의 특수문자로 시작되는 파일이름을 바꾸려면 다음과 같이 한다.

```
mv  ./bad_filename  new_filename
mv  ./?bad_filename  new_filename
mv  ./*bad_filename  new_filename
```

등록부 sourcedir 와 이 등록부의 내용을 그 파일체계의 새로운 위치에 있는 targetdir 로 이동시키자면 다음과 같이 하여야 한다.

```
mv  sourcedir  targetdir
```

현재등록부에서 모든 파일 및 등록부들의 련결도 포함하여 targetdir 로 이동시키자면 다음과 같이 하여야 한다.

mv * targetdir

sourcedir 등록부에 있는 모든 파일 및 등록부들의 련결도 포함하여 targetdir 로 이동시키자면 다음과 같이 하여야 한다(sourcedir 와 targetdir 는 서로 다른 등록부경로에 있다).

mv sourcedir/* targetdir

경고

만일 file1 과 new_file 이 서로 다른 파일체계에 존재한다면 mv 는 그 파일을 복사하고 원래의것을 제거한다. 이 경우에 이동시키는 사람이 소유자로 되며 다른 파일들과의 모든 련결관계는 없어 진다. 즉 mv 는 파일체계들사이의 련결까지는 운반하지 못한다. 만일 file1 이 등록부라면 mv 는 전체 등록부구조를 목적파일체계에 복사하고 원래의것을 제거해 버린다.

mv 는 다음과 같은 연산을 수행할수 없다.

- 현재작업등록부나 그것의 어미등록부의 이름을 .과 ..을 리용하여 바꾸는 연산
- 동일한 어미등록부에 포함되는 꼭 같은 이름으로 등록부의 이름을 새롭게 바꾸는 연산

종속성

NFS
망파일의 접근조종목록들은 stat()에 의한 st_mode 로 귀환될 때 집약되지만 새로운 파일에 복사되지는 않는다. 이런 파일들에 대하여 mv 혹은 ln 지령을 리용하는 경우에 파일의 겹쳐쓰기를 확인할 때 방식값뒤에 +는 출력되지 않는다.

저자

mv 는 AT&T 와 캘리포니아종합대학, 버클리, HP 에서 개발하였다.

관련항목

cp(1), cpio(1), ln(1), rm(1), link(1M), lstat(2), readlink(2), stat(2), symlink(2), symlink(4), acl(5)

표준일치

mv: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

pwd

pwd - 현재작업등록부이름을 귀환한다.

---

pwd(1)

이름

pwd - 현재작업등록부이름을 귀환한다.

형식

pwd

해설

pwd는 현재의 작업등록부의 경로이름을 출력한다.

외부적영향

환경변수들

LC_MESSAGES는 통보가 현시되는 언어를 결정한다.

만일 LC_MESSAGES가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG의 값이 기정값으로 리용된다. LANG이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 pwd는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

진단

Cannot Open ..

Read error in ..

가능한 파일체계가 고장이다. 체계관리자에게 문의해야 한다.

pwd: cannot access parent directories

현재등록부가(보통 다른 과제에 의하여) 제거되였다. cd 지령을 리용하여 유효한 등록부로 이동하여야 한다(cd(1)을 참고).

실례

pwd지령은 현재등록부의 경로를 표시한다. 만일 자기의 홈등록부가 /mnt/staff이고

cd camp/nevada 를 홈등록부에서 수행하였다면 pwd 는 다음과 같이 표시한다.

/mnt/staff/camp/nevada

저자

mv 는 AT&T 와 HP 에서 개발하였다.

관련항목

cd(1)

표준일치

mv: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

rm

rm - 파일 및 등록부들을 제거한다.

---

rm(1)

이름

rm - 파일 및 등록부들을 제거한다.

형식

rm [-f | -i] [-Rr] file …

해설

rm 은 등록부로부터 한개이상의 파일들을 제거한다. 파일의 제거는 그 등록부에 대한 쓰기와 탐색(실행)허락을 요구하지만 파일 그자체에 대한 허락은 요구하지 않는다. 그러나 그 파일을 포함하는 등록부에 대한 보호비트가 설정되여 있으면 그 파일의 소유자, 등록부의 소유자 또는 특수권한을 가진 사용자만이 그 파일을 제거할수 있다.

만일 사용자가 제거하는 파일에 대한 제거허락을 가지고 있지 못하고 표준입력장치가 말단이라면 그 파일의 이름과 허락을 포함하면서 파일의 제거를 확인하는 통보가 나온다(아래의 접근조종항목을 참고). y 로 대답하면 파일은 제거된다. 그러나 -f 를 리용하면 아무런 질문도 제기하지 않는다.

file 이 등록부이고 -f 가 규정되지 않았으며 파일에 대한 쓰기허락이 허용되지 않고 표준입력장치가 말단이거나 -i 가 규정되였다면 rm 은 표준오유장치에 통보를 내보내고 표준입력장치로부터 대답을 받는다. 대답이 yes 가 아니면 아무런 작용도 하지 못한다.

선택항목들

- -f 모든 파일이나 등록부가 file 의 허락에 무관계하게 아무런 확인도 없이 제거되도록 한다. 이 선택항목은 존재하지 않는 대상을 제거하려고 해도 진단통보를 내보내지 않는다. 그러나 기타 경우에 대한 진단통보는 금지하지 않는다. 아무런 통보도 내보내지 않도록 하려면 -f 가 표준오유출력장치를 /dev/null 로 방향바꾸기하여 리용되여야 한다. 이 선택항목은 이전의 -i 를 무시한다.

- -i 이 선택항목은 매개 대상을 제거하기전에 그 제거를 확인하는 통보를 내보낸다. 이 선택항목은 이전의 -f 를 무시한다.

- -R 제거대상이 등록부인 경우에는 그 등록부자체를 제거하기전에 등록부의 전체 내용을 재귀적으로 제거하도록 한다. -i 와 결합하여 매개 등록부를 제거하기전에 사용자에게 대화식으로 문의한다.

- -r 이 선택항목은 -R 와 동등하다.

접근조종목록(ACL)

만일 파일이 ACL 을 선택적으로 가진다면 rm 은 파일의 허락뒤에 +를 현시한다. 허락들은 stat()가 귀환하는 파일의 st_mode 값을 합친것이다(stat(2)과 acl(5)를 참고).

외부적영향

환경변수들

LANG 은 설정되지 않았거나 null 인 국제화변수에 대한 기정값을 제공한다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)을 참고). 어떤 국제화변수가 틀린 설정을 포함하면 rm 은 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 비지 않은 기호렬값으로 설정되여 있으면 기타 국제화변수들의 모든 값을 무시한다.

LC_CTYPE 는 파일이름의 단일/다중바이트기호들, 인쇄가능한 기호들의 분류, 정규식에서 기호클라스식들에 대한 해석을 결정한다.

LC_MESSAGES 는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용과 표준적인 출력장치에 씌여 지는 비형식적인 통보에 영향을 주는 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

진단

일반적으로는 자체로 설명한다(-f 가 진단통보모두를 금지하지 않는다는것을 참고).

아래와 같은 지령을 사용하는것을 피하기 위하여 파일 ..으로 제거하는것은 좋지 않다.

```
rm -r .*
```

목표대상이 등록부라면 -R 혹은 -r 가 없을 때 오유통보가 현시된다.

실례

대화식으로 파일들을 제거하려면 다음과 같이 하여야 한다.

```
rm -i file1 file2
```

등록부의 모든 파일들을 제거하려면 다음과 같이 하여야 한다.

```
rm -i mydirectory/*
```

우의 실례에서 mydirectory 안에 있는 파일들만 제거하고 보조등록부는 제거하지 않는다는것을 주의하여야 한다.

현재등록부에서 -, * 혹은 기타 쉘의 특수기호로 표시되는 파일들을 제거하려면 다음과 같이 하여야 한다.

```
rm ./-filename
rm ./*filename
```

등

현재등록부에서 어떤 이상한 기호로 시작되거나 앞뒤에 공백이 있는 파일들에 대하여 그것들을 확인하면서 제거하려면 다음과 같이 하여야 한다.

```
rm -i *filename*
```

등록부를 지우는 강력하고도 위험한 지령은 다음과 같다.

```
rm -fR directoryname
```

혹은

```
rm -Rf directoryname
```

이 지령은 파일이나 등록부를 제거할 때 아무런 확인도 없이 directoryname 의 모든 파일과 등록부들을 제거한다. 따라서 이 지령은 directoryname 까지도 포함하여 그안의 모든 파일들과 보조등록부들을 지우는 경우에만 사용하여야 한다.

종속성

NFS
망파일을 제거하기전에 그 확인을 문의할 때 선택적인 접근조종목록의 존재성을 가리키는 +를 현시하지 않는다.

관련항목

rmdir(1), unlink(2), acl(5)

표준일치

rm: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

rmdir

rmdir - 등록부들을 제거한다.

---

rmdir(1)

이름

rmdir - 등록부들을 제거한다.

형식

rmdir [-f | -i] [-p] dir …

해설

rmdir는 빈 등록부를 제거한다.
등록부들은 지정된 순서로 제거된다. 따라서 등록부와 그것의 보조등록부들을 인수로 지정하면 보조등록부가 어미등록부보다 먼저 규정되여야 한다. rmdir는 어미등록부가 빈 다음에야 그것을 제거한다. 등록부제거는 그 등록부의 어미등록부에 대한 쓰기와 탐색(수행)허락을 요구한다. 등록부자체에 대한 허락은 아무것도 요구하지 않는다. 그러나 보호비트가 어미등록부에 설정되여 있으면 등록부소유자, 어미등록부소유자, 특수권한을 가진 사용자만이 그 등록부를 제거할수 있다.

선택항목들

-f　　-i가 있어도 관계없이 아무런 문의도 없이 파일들을 제거한다. 이 선택항목은 존재하지 않는 인수들에 대한 진단통보도 금지한다. 그러나 기타 경우에 대한 진단통보는 금지하지 않는다. 아무런 통보도 내보내지 않도록 하려면 -f가 표준오유출력장치를 /dev/null로 방향바꾸기하여 리용되여야 한다. 이 선택항목은 이전의 -i를 무시한다.

-i　　이 선택항목은 매개 등록부를 제거하기전에 그것을 확인하는 통보를 내보낸다. 이 선택항목은 이전의 -f를 무시한다.

-p　　만일 한개이상의 경로이름으로 된 등록부를 제거한후에 그

것의 어미등록부가 비게 되면 rmdir 는 그 어미등록부까지도 제거한다. 이 과정은 비지 않은 어미등록부가 나타날 때까지 계속된다. -i 와 함께 리용하면 rmdir 는 지울 때 매개에 대하여 확인을 하면서 제거한다.

외부적영향

환경변수들

LANG 은 설정되지 않았거나 null 인 국제화변수에 대한 기정값을 제공한다. LANG 이 결정되지 않았거나 null 이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 rmdir 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 비지 않은 기호렬값으로 설정되여 있으면 기타 국제화변수들의 모든 값을 무시한다.

LC_CTYPE 는 dir 이름의 단일/다중바이트기호들, 인쇄가능한 기호들의 분류, 정규식에서 기호클라스식들에 대한 해석을 결정한다.

LC_MESSAGES 는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용과 표준적인 출력장치에 씌여 지는 비형식적인 통보에 영향을 주는 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

진단

일반적으로는 자체로 설명한다. -f 가 진단통보모두를 금지하지 않는다는것을 주의하여야 한다.

실례

대화식으로 등록부들을 제거하려면 다음과 같이 하여야 한다.

```
rmdir -i directories
```

가능한껏 많은 경로를 제거하려면 다음과 같이 하여야 한다.

```
rmdir -p component1/component2/dir
```

관련항목

rm(1), rmdir(2), stat(2)

표준일치

rmdir：SVID2, XPG2, XPG3, XPG4

제4장. 파일보기 - 방향바꾸기, cat, more, pg, head, tail 지령

이 장에서는 몇가지 각이한 파일보기방법들을 고찰함으로써 UNIX 지령들을 가지고 보다 심도 있는 작업을 하기 위한 준비를 갖추기로 한다. 먼저 몇가지 공통적인 방향바꾸기수법들을 고찰하고 파일보기와 관련한 지령들을 보겠다. 이 장에서 고찰하는 내용은 다음과 같다.

- 방향바꾸기
- cat, more, pg, head, tail 지령들

방향바꾸기

UNIX 는 보통 표준입력장치라고 부르는 건반으로부터 지령들을 받아서 표준출력장치라고 부르는 화면에로 그 결과를 내보내도록 설치된다. 지령들은 오유통보도 화면으로 보낼수 있다. 그러나 항상 자료의 입력이 표준입력장치로부터 들어 와야 하며 결과 및 오유는 표준출력장치로 나가야 한다는 법은 없다. 사용자는 이 설정을 조종할수 있다. 이것을 **방향바꾸기**(Redirection)라고 부른다. 표 4-1 은 여러가지 공통적인 방향바꾸기형식들을 보여 주고 있다.

표에서 보여 준바와 같이 지령의 결과를 표준출력으로부터 파일로 바꾸기 위해서는 기호 ">"를 리용한다. 만일 사용자가 환경변수 noclobber 를 설정하였다면 이미 존재하는 파일에로의 방향바꾸기는 진행되지 않는다. 즉 noclobber 는 이미 존재하는 파일우에 덧쓰는 방향바꾸기를 허용하지 않는다. 아래의 실례에서와 같이 /tmp/processes 라는 이미 존재하는 파일우에 쓰려고 시도하면 파일이 존재한다는 통보를 받게 된다.

\# **ps -ef > /tmp/processes**

/tmp/processes**:** File exists

그러나 강제로 덧쓰기하면서 방향바꾸기를 하려면 기호 "!"를 리용한다. ">!"은 파일을 덧쓰기하며 기호 ">>!"는 이미 존재하는 파일뒤에 결과가 강제적으로 추가되게 한다. 이러한 실례들을 표 4-1 에서 주고 있다.

표 4-1 공통적으로 사용되는 방향바꾸기형식들

<table>
<tr><th>지 령</th><th>실 례</th><th>해 설</th></tr>
<tr><td><</td><td>Wc –l < .login</td><td>표준입력장치방향바꾸기:
wc(word count)를 수행하고 .login 안의 행의 개수를 렬거한다.</td></tr>
<tr><td>></td><td>ps –ef > /tmp/processes</td><td>표준출력장치방향바꾸기:
ps 를 수행하고 그 결과를 /tmp/processes 파일에 보낸다.</td></tr>
<tr><td>>></td><td>ps –ef >> /tmp/processes</td><td>표준출력장치에 추가:
ps 를 수행하고 그 결과를 /tmp/processes 뒤에 추가한다.</td></tr>
<tr><td>>!</td><td>ps –ef >! /tmp/processes</td><td>표준출력장치에 추가 및 noclobber 를 무시:
/tmp/processes 가 이미 존재하여도 겹쳐 쓴다.</td></tr>
<tr><td>>>!</td><td>ps –ef >>! /tmp/processes</td><td>표준출력장치에 추가 및 noclobber 를 무시:
/tmp/processes 뒤에 추가한다.</td></tr>
<tr><td>|(파이프)</td><td>ps | wc -l</td><td>ps 를 수행하고 그 결과를 wc 의 입력으로 리용한다.</td></tr>
<tr><td>0-표준입력장치
1-표준출력장치</td><td></td><td></td></tr>
<tr><td>2-표준오유장치</td><td>cat program 2 > errors</td><td>프로그람파일을 표준출력장치에 결합시키고 오유는 errors 파일로 방향바꾸기한다.</td></tr>
<tr><td></td><td>cat program 2 >> errors</td><td>프로그람파일을 표준출력장치에 결합시키고 오유는 errors 파일뒤에 추가한다.</td></tr>
<tr><td></td><td>find /-name '*.c' -print > cprograms 2 > errors</td><td>체계에서 .c 로 끝나는 모든 파일들을 탐색하고 현재작업등록부안에 있는 cprograms 에 그 파일들의 목록을 출력하며 모든 오유(파일서술자 2)는 현재작업등록부의 errors 파일에 보낸다.</td></tr>
<tr><td></td><td>find /-name '*.c' -print > cprograms 2 > &1</td><td>체계에서 .c 로 끝나는 모든 파일들을 탐색하고 현재작업등록부안에 있는 cprograms 에 그 파일들의 목록을 출력하며 모든 오유(파일서술자 2)는 현재작업등록부의 파일서술자 1(cprograms)과 같은 장소에 보낸다.</td></tr>
</table>

표에서 보여 준 기호들을 리용하여 표준입력장치와 표준출력장치를 방향바꾸기할수 있다. 실례로 화면이 아닌 다른 출력장치인 파일에로 지령의 결과를 보낼수 있다.

cat, more, pg, head, tail을 리용한 파일의 보기

긴 파일을 화면에서 보려고 하는 경우에 그 내용이 너무 많아서 화면의 크기를 벗어나는 경우가 있을수 있다.

cat(concatenate의 략자)지령은 파일이 긴 경우에 그 파일의 끝부분만 화면에 남긴다. 이미 앞장에서 고찰한 denise 사용자의 등록부에 있는 파일들을 렬거하되 다음의 지령에서처럼 listing이라는 파일로 방향바꾸기를 하여 보자.

$ ls -a /home/denise > listing

그림 4-1에서처럼 cat listing 지령을 수행시키면 그 파일의 끝부분만 화면에서 볼수 있다. cat지령은 -n을 가질수 있는데 여기서 n은 행번호이다. 즉 n을 사용하면 그 행번호로부터 파일의 끝까지 볼수 있다.

```
cat -n example
106  rclock.exe
107  rkhelp.exe
108  sb.txt
109  shellupd.exe
110  smsup2.exe
111  softinit.remotesoftcm
112  srVpr.exe
113  tabnd1.exe
114  target.exe
115  tcp41a.exe
116  tnds2.exe
117  trace.TRC1
118  trace.TRC1.Z.uue
119  upgrade.exe
120  uue.syntax
121  v103.txt
122  whoon
123  win95app.exe
124  wsdrv1.exe
125  wsos21.exe
126  wsos22.exe
127  wsos23.exe
128  xferp110.zip
$
```

그림 4-1. 지령 cat -n

파일의 끝부분만 보는것은 의의가 없으므로 pg(page의 략자)지령을 사용하여 그림 4-2에서처럼 한번에 한개 화면씩 볼수 있다.

```
pg example
.
..
.CbtOptSet
.Xauthority
.cshrc
.cshrc.orig
.elm
.exrc
.fmrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
:
```

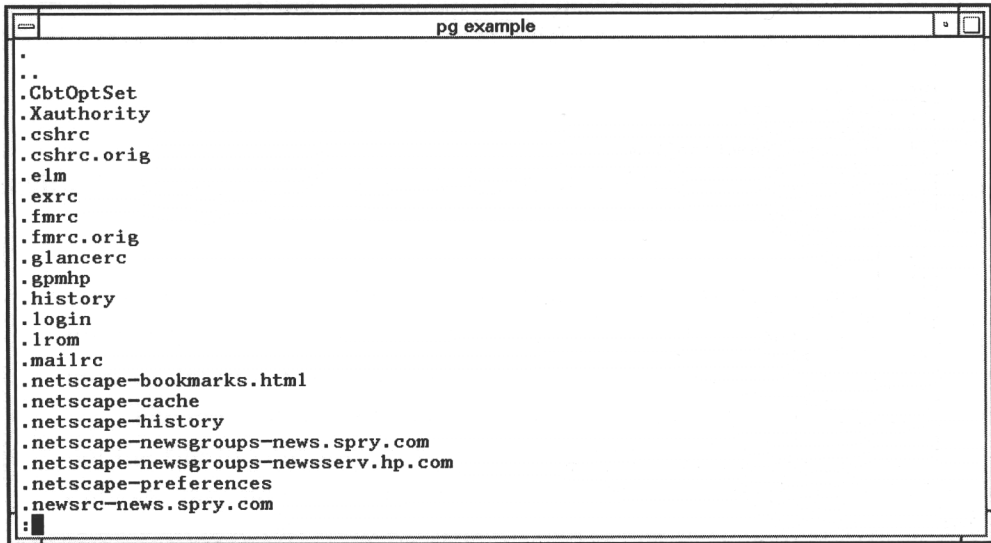



그림 4-2. 지령 pg

more 지령은 그림 4-3에서 보여 준것처럼 pg와 동일한 결과를 만든다.

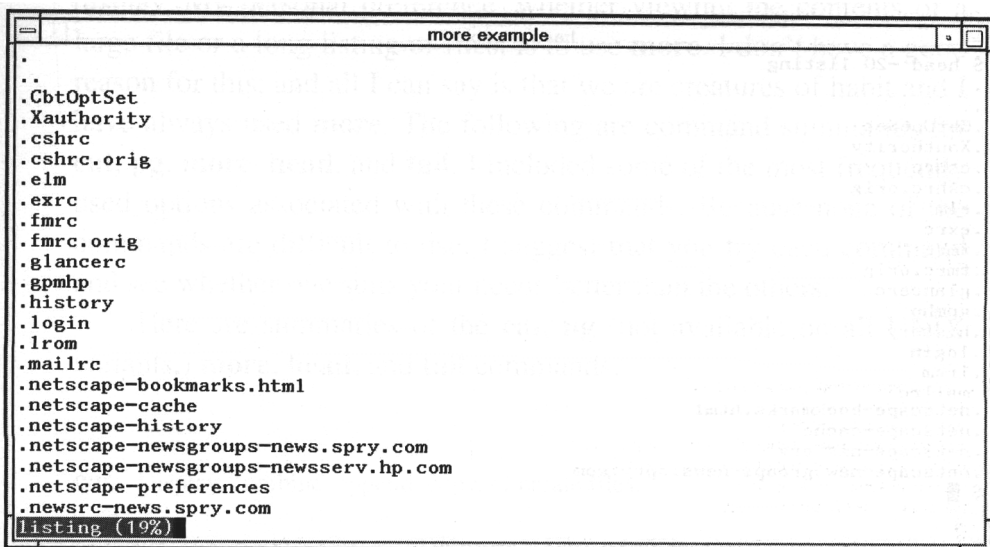
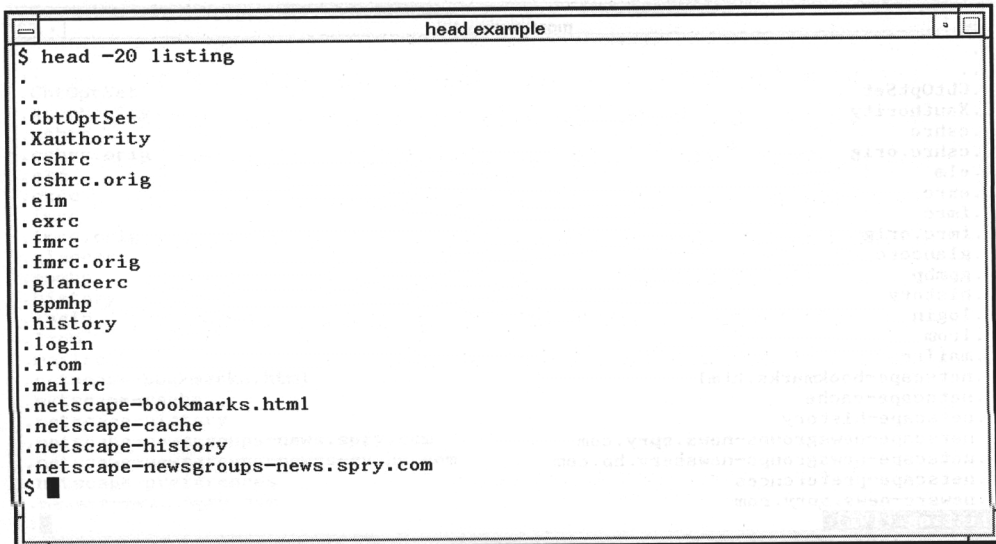
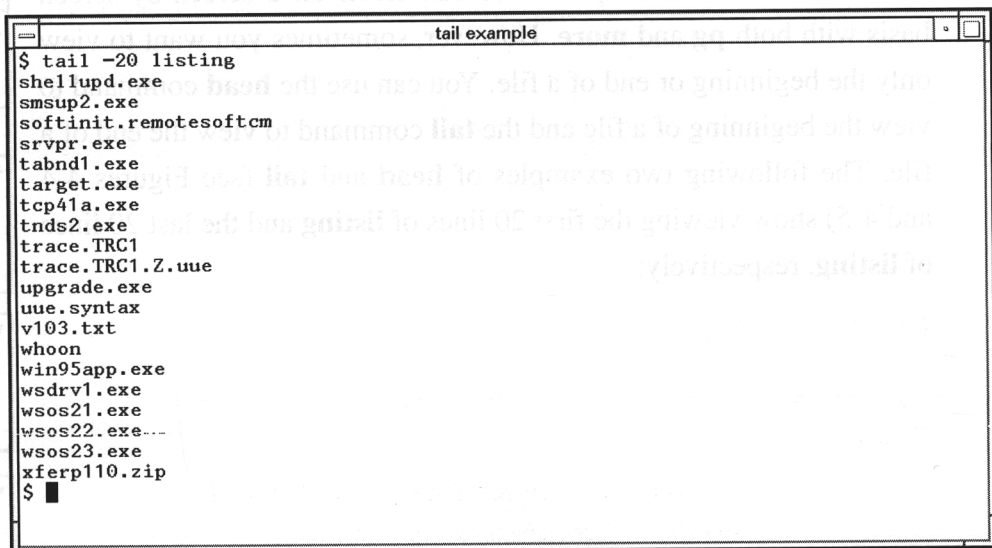



그림 4-3. 지령 more

이렇게 pg 및 more 지령을 둘 다 사용하여 한개 화면씩 흐르게 할수 있지만 때때로 파일의 시작이나 끝만을 보고 싶을수도 있다. head 지령은 파일의 시작부분을 보여 주며 tail 지령은 파일의 끝부분을 보여 준다. 다음의 두개의 실례(그림 4-4 와 그림 4-5)는 head와 tail을 사용하여 listing의 첫 스무개 행과 마지막 스무개 행을 각각 보여 주고 있다.

```
head example
$ head -20 listing
.
..
.CbtOptSet
.Xauthority
.cshrc
.cshrc.orig
.elm
.exrc
.fmrc
.fmrc.orig
.glancerc
.gpmhp
.history
.login
.lrom
.mailrc
.netscape-bookmarks.html
.netscape-cache
.netscape-history
.netscape-newsgroups-news.spry.com
$ █
```

그림 4-4. 지령 head

```
tail example
$ tail -20 listing
shellupd.exe
smsup2.exe
softinit.remotesoftcm
srvpr.exe
tabnd1.exe
target.exe
tcp41a.exe
tnds2.exe
trace.TRC1
trace.TRC1.Z.uue
upgrade.exe
uue.syntax
v103.txt
whoon
win95app.exe
wsdrv1.exe
wsos21.exe
wsos22.exe
wsos23.exe
xferp110.zip
$ █
```

그림 4-5. 지령 tail

이러한 지령들의 사용은 보려는 파일의 정보에 의존한다. 가령 크거나 긴 파일의 내용을 볼 때에는 more 지령을 즐겨 사용하는 사람들이 많다. 아래에서는 cat, pg, more, head, tail 지령들을 개괄하고 있다. 이 지령들은 대단히 사용하기가 쉽기때문에 그중에서 적당한것을 골라 쓰면 된다.

cat − 파일을 현시, 결합, 추가, 복사, 창조한다.

선택항목들

- `-` 표준입력장치로서 건반을 리용하려고 할 때 리용된다.
- `-n` 행번호들이 출력행들과 함께 현시된다.
- `-p` 여러개의 련속적인 빈 행들이 하나의 빈 행으로 바뀐다.
- `-s` 존재하지 않는 파일에 대한 정보를 금지하는 "고요한" 방식이다.
- `-u` 출력은 완충화되지 않고 바이트별로 조종된다. 대부분의 인쇄불가능한 문자들도 보이도록 인쇄한다.

pg − 파일을 전부 혹은 부분적으로 현시한다.

선택항목들

- `-number` number는 현시하려는 행들의 개수를 지적한다.
- `-p string` string은 대화식통보문으로 리용된다.
- `-c` 파일의 다음페지를 현시하기전에 화면을 지운다.
- `-f` 현시되는 행들을 자르지 않는다.
- `-n` 새 행문자를 만들지 않고 지령글자가 입력되자마자 지령이 만들어지게 한다.

more − 한번에 한개 화면씩 파일의 전부 혹은 부분을 현시한다.

선택항목들

- `-c` 파일의 다음페지를 현시하기전에 화면을 지운다.
- `-d` 화면의 아래에 간단한 지령을 대화식으로 내보낸다.
- `-f` 본문을 화면의 크기에 맞게 자르고 이에 따라 페지의 길이를 조절한다.
- `-n` 현시창문의 행 개수를 n으로 설정한다.
- `-s` 여러개의 련속적인 빈 행들을 하나의 빈 행으로 바꾼다.

head − 파일의 처음 몇개 행만 내보낸다.

선택항목들

-c	출력은 규정된 개수의 바이트만큼 생성된다.
-l	출력은 규정된 개수의 행만큼 생성된다. 이것은 기정값이다.
-n count	count에 의하여 바이트개수 혹은 행 개수가 규정된다. -count로 리용할수도 있다. count의 기정값은 10이다.

tail - 파일의 마지막 몇개 행만 내보낸다.

선택항목들

-b number	파일의 끝으로부터 현시를 시작하려는 블로크의 개수를 규정한다.
-c number	파일의 끝으로부터 현시를 시작하려는 문자들의 개수를 규정한다.
-n number	파일의 끝으로부터 현시를 시작하려는 행들의 개수를 규정한다.

지 령 소 개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 좀 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

cat

cat - 파일들을 련결한다.

---

cat(1)

이름

cat - 파일들을 련결, 복사, 출력한다.

형식

cat [-benrstuv] file …

해설

cat는 매개 파일을 순차적으로 읽어서 그것을 표준출력장치에 쓴다.
다음의 형식은 파일을 기정으로 표준출력장치에 쓴다.

cat file

다음의 형식은 file1과 file2를 련결하여 그 결과를 file3에 넣는다.

cat file1 file2 > file3

파일인수로서 -가 나타나면 cat는 표준입력장치를 리용한다. 표준입력장치와 다른 파일들을 결합하기 위해서는 결합기호 -로 파일인수들을 련결한다.

선택항목들

-b -n이 규정되였을 때 빈 행들에서 행번호들을 없애버린다. 이 선택항목은 규정되면 -n이 자동적으로 설정된다.

-e 매개 행의 끝에 $기호를 출력한다. 이 선택항목은 규정되면 -v가 자동적으로 설정된다.

-n 1부터 순차적으로 앞에 행번호를 붙여 행들을 출력한다.

-r 여러개의 련속적인 빈 행들을 하나의 빈 행으로 바꾼다. 그리하여 기호들을 포함하는 행들사이에 빈 행이 한개밖에 없도록 한다.

-s cat는 파일이 존재하지 않는것, 동일한 입력과 출력장치, 쓰기에 대한 오유통보를 모두 금지한다. 이것을 **Slient 선택항목**(Slient Option)이라고 부른다.

-t 매개 태브기호를 ^I로 출력한다. 이 선택항목은 규정되면 -v가 자동적으로 설정된다.

-u 출력이 완충화되지 않는다. 즉 문자별로 조종된다. 그러나 출력은 완충화되는것이 보통이다.

-v 인쇄불가능한 기호들(태브, 행바꾸기기호의 입장 등을 제외하고)이 보이게 인쇄되도록 한다. 조종기호들은 ^X(Ctrl-X)형식으로 인쇄되며 DEL 기호(8진수 0177)는 ^?로 인쇄된다(ascii(5)를 참고). 가장 중요한 비트가 설정된 단일바이트조종기호들은 M-^x 형식으로 인쇄된다. 여기서 x는 7개의 낮은 준위의 비트에 의하여 규정되는 기호이다. 기타 인쇄불가능한 기호들은 M-x 형식으로 인쇄된다. 이 선택항목은 LC_CTYPE 환경변수와 그것이 대응하는 코드모임의 영향을 받는다.

외부적영향

환경변수들

LANG 은 규정되지 않았거나 null 인 국제화변수에 기정값을 제공한다. 만일 LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 cat 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모든 값을 무시한다.

LC_CTYPE 는 본문의 단일/다중바이트기호들, 인쇄가능한 기호들의 분류, **정규식**(Regular Expression)에서 기호모임식들에 대한 해석을 결정한다.

LC_MESSAGES 는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용의 위치와 표준적인 출력장치에 씌여 지는 비형식적인 통보의 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

귀환값

다음의 값들중의 하나로 귀환한다.

0	파일들이 성과적으로 련결되였다.
>0	오유가 발생하였다. 파일들은 련결되지 않았다.

실례

길이가 령인 파일을 창조하려면 다음의 지령들중에서 하나를 리용한다.

```
cat /dev/null > file
cp /dev/null file
touch file
```

다음의 지령은 file1 에 있는 모든 태브기호들에 대하여 ^I 를 인쇄한다.

```
cat -t file1
```

존재하지 않는 파일들에 대하여 오유통보가 나오는것을 금지하기 위해서는 다음의 지령을 리용한다.

```
cat -s file1 file2 file3 > file
```

만일 file2 이 존재하지 않으면 우의 지령은 file2 에 대하여 오유통보를 내보내지 않고 file1 과 file3 을 련결시킨다. 오유통보를 현시하지 않는것을 제외하고는 -s 를 리용하지 않았을 때와 결과는 꼭 같다.

다음의 지령은 file2에서 인쇄불가능한 기호들을 보여 준다.

cat -v file2

경고

cat file1 file2 > file1 과 같은 지령은 파일들을 련결하기전에 file1 에 자료들을 덧쓰기하기때문에 파일을 파괴시킨다. 그러므로 쉘의 특수기호들을 리용할 때 주의해야 한다.

관련항목

cp(1), more(1), pg(1), pr(1), rmnl(1), ssp(1)

표준일치

cat: SVID2, SVID3, XPG2, XPG4, POSIX.2

head

head – 파일의 처음 몇개 행만을 출력한다.

---

head(1)

이름

head – 처음 몇개 행만을 출력한다.

형식

head [-c | -l] [-n count] [file …]

절대형식:

head [-count] [file …]

해설

head 지령은 규정된 매개 파일들 혹은 표준입력장치에서 처음부터 count 만한 개수의 행들을 표준출력장치에 인쇄한다. 만일 count 가 생략되였으면 기정값으로 10을 취한다.

여러개의 파일들이 규정되면 head 는 매 파일의 앞에 다음과 같은 형식의 행을 출력한다.

==> file <==

선택항목들

-c　출력량이 바이트로 측정된다.

-count　출력단위들의 개수를 지정한다. 이 선택항목은 이전과의 호환성을 보장하기 위하여 제공된다(아래의 -n 을 참고). 이 인수는 다른 모든 선택항목들과는 배타적으로 규정된다.

-l　출력량이 행으로 측정된다. 이 선택항목은 기정으로 설정된다.

-n count

출력되는 행개수(기정값) 또는 바이트의 개수를 지정한다. count 는 부호 없는 10 진수이다. 만일 -n(혹은 -count)이 주어지지 않으면 기정값은 10 이다. 이 선택항목은 -count 와 같은 기능을 수행하지만 보다 더 표준적인 방법이다. -n 의 사용은 체계들사이의 이동성이 중요한 곳에서 자주 리용된다.

외부적영향

환경변수들

LC_CTYPE 는 단순한 and/or 다중문자바이트로 파일안에 있는 본문에 대한 설명을 결정한다.

LC_MESSAGES 는 통보가 현시되는 언어를 결정한다.

만일 LC_CTYPE 혹은 LC_MESSAGES 가 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 대신에 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 head 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임의 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

관련항목

tail(1)

표준일치

head: SVID3, XPG4, POSIX.2

more

more – 한번에 한개 화면씩 파일의 전부 혹은 부분을 현시한다. 즉 파일을 려파시켜 보는 지령이다.

---

more(1)

이름

more, page – crt 보기에 대한 파일읽기려파기이다.

형식

more [-n] [-cdefisuvz] [-n number] [-p command] [-t tagstring] [-x tabs] [-W option] [+linenumber] [+/pattern] [name …]

e [-n] [-cdefisuvz] [-n number] [-p command] [-t tagstring] [-x tabs] [-W option] [+linenumber] [+/pattern] [name …]

주의:

pg 지령이 많이 리용되고 몇가지 기능들이 추가도 되였지만 pg 지령에서는 문자의 강조표시기능은 제공하지 못하고 있다(pg(1)을 참고).

해설

more 는 한번에 한 화면씩 본문을 련속적으로 검사하는 려파기로서 프로그람으로 작성된 말단장치에서 작용한다. 이것은 pg 와 대단히 류사하며 이전 방안과의 호환성이 있다. more 는 표준적으로 파일에서 한개의 화면을 출력한후에 화면밑에 그 파일이름을 현시한다. 여기서 한 행을 더 현시하려면 <Enter>건을 누르고 한개 화면을 더 현시하려면 <Space>건을 누른다. 다른 가능성들은 아래에서 설명한다.

more 와 pg 는 약간 다르다. more 는 다음 페지를 출력할 때 화면을 우로 흘러 보낸다. 그러나 pg 는 그 화면을 지우고 다음페지의 새로운 화면을 출력한다. 두개의 지령이 다 화면들 호상간에 한개의 겹친 행을 가진다.

name 은 파일이름이거나 표준입력장치를 규정하는 -가 될수 있다. more 는 파일인수들을 주어 진 순서대로 처리한다.

more 는 **기본정규식문장론**(Basic Regular Expression Syntax)을 보장한다(regexp(5)를 참고).

선택항목들

-n number

현시창문의 행개수를 정인 10 진수로 규정한다. 기정값은 말단에서 현시되는 행개수보다 1 만큼 작다. 즉 화면은 24 개 행을 현시하므로 기정값은 23 이다. 이 선택항목은 환경으로부터 얻어 지는 모든 값들을 무시한다.

-n 행개수가 n 으로 설정된다. 우의것과 동일하다.

-c 화면의 꼭대기에서부터 시작하여 매 페지를 현시하는데 매개행은 페지를 현시하기 직전에 지운다. 즉 화면이 흘러 가는 것을 막고 다음화면을 쉽게 읽을수 있게 해준다. 이 선택항목은 말단장치가 행의 끝까지 지우는 기능성이 없으면 무시된다.

-d 대화식으로 통보를 내보내며 계속하는 경우에는 <Space>건을, 탈퇴는 <q>건을, 도움말은 <h>건을 누르도록 한다.

-e 마지막파일의 마지막행을 출력한후에 즉시 탈퇴한다.

-f 화면에서의 물리적행이 아니라 론리적행을 인식한다. 즉 긴 행은 다 볼수 없다. 이 선택항목은 nroff 출력이 ul 로 관통(파이프)될 때 효과적이다. 왜냐하면 ul 이 **탈퇴렬**(Escape Sequences)을 생성할수 있기때문이다. ul 이 포함하는 문자들은 초기에는 화면위치를 차지하지만 그것들은 말단장치로 보내여 질 때 출력되지 않는다. 이렇게 more 는 행들이 실지로는 보다 더 길다고 가정하며 그것들을 오유없이 받아 들인다.

-i 대소문자의 구별이 없이 패턴정합탐색을 진행한다.

-s 여러개의 빈 행들을 출력할 때 하나의 빈 행으로 바꾼다. 이것은 특히 nroff 출력을 볼 때 효과적이다.

-u 일반적으로 특수말단장치의 고유한 방식으로 nroff 에서 생성되는것과 같은 밑선 및 강조체양식을 조종한다. 만일 말단장치가 밑선을 보장하거나 강조표시방식(보통 반전비데오)을 가진다면 more 는 원천파일의 밑선정보에 대하여(강조표시방식은 보장하지 않고) 밑선을 허용하는 탈퇴렬을 출력하게 한다. 만일 말단이 강조표시를 보장한다면 more 는 강조형으로 인쇄되여야 한다는 방식정보를 리용한다. 선택항목 -u 는 "ul" 및 "os"말단정보기발들의 이 처리를 무시한다.

-v 인쇄불가능한 기호들을 도형으로 현시하지 않는다. 모든 비 ASCII 문자 및 조종기호들(<Tab>, <Backspace>, <Enter>는 제

외)은 기정으로 Ctrl-X 는 ^X, 비 ASCII 기호 x 는 M-x 형식으로 눈에 보이게 현시된다.

-z　　<Backspace>는 ^H, <Enter>는 ^M, <Tab>는 ^I 를 현시한다는 것을 제외하고는 -v 를 규정하지 않았을 때와 같다.

-p command

more 는 지령인수에서 매개 검사되는 파일보다 command 를 먼저 수행한다. 만일 command 가 행번호 혹은 표준식탐색과 같은 위치를 지정하는 지령이라면 현재위치를 파일의 중간행들이 아니라 command 의 최종결과를 표시하도록 설정한다. 이런 위치를 지정하는 command 가 성과적으로 수행되면 파일의 첫행이 현재위치로 된다.

-t tagstring

tagstring 으로 규정된 **태그**(Tag)를 포함하는 파일의 한개 화면부분을 현시한다. 규정된 태그는 현재위치에서 나타난다. -p 및 -t 가 둘 다 규정되면 more 는 -t 를 먼저 처리한다.

-x tabs

매 태브위치에 태브점을 설정한다. tabs 의 기정값은 8 이다.

-W option

다음과 같은 more 지령의 확장내용을 선택적으로 제공한다.

notite　　말단의 초기화기호렬을 파일을 현시하기전에 보내지 않도록 한다. 또한 말단의 초기화해제기호렬을 탈퇴전에 보내지 않도록 한다.

tite　　초기화 및 초기화해제기호렬들을 보내도록 한다. 이것은 기정값이다.

+linenumber

현재위치가 linenumber 로 설정되도록 현시를 시작한다.

+/pattern

현재위치가 주어 진 패턴을 들여 맞추는 행에 설정되도록 현시를 시작한다.

화면에서 가능한 행개수는 -n 에 의하여 결정된다. 그런데 실지로 씌여 지는 행개수는 1 만큼 작다. 왜냐하면 화면의 마지막행이 사용자와의 대화식통보문을 쓰는데 리용되기때문이다.

한 행에 가능한 렬의 개수는 환경값으로 결정된다. more 는 론리적행을 인식하여

렬의 개수보다 더 많은 기호들을 포함하는 행을 출력한다. 론리적행들은 서로 독립적으로 씌여 지며 지령들은 그것을 서로 분리시켜 처리한다.

행개수와 렬개수를 결정할 때 우에서 서술한 방법들이 적용되지 않으면 more 는 terminfo 서술자파일을 리용한다(term(4)를 참고). 이것도 실패하면 기정값은 각각 24 및 80 으로 설정된다.

표준출력장치가 말단이고 -u 가 규정되지 않았을 때 more 는 <Backspace>기호와 <Enter>기호를 특수하게 처리한다.

<Backspace>기호가 작용된 기호는 밑선이 붙은 기호로 나타나게 한다.
두개의 동일한 인쇄가능한 기호들사이에 나타나는 <Backspace>기호는 그 두 기호들중에서 첫문자가 강조체로 나타나도록 하며 두번째 기호는 사라지게 한다.
기타 <Backspace>기호렬은 말단에 직접 씌여 진다. 이때 <Backspace>기호앞에 있는 기호는 표시되지 않는다.
행 끝에 있는 <Enter>기호는 조종기호로 씌여 지지 않고 무시된다.

만일 표준출력장치가 말단장치가 아니라면 more 는 항상 인수목록에 있는 마지막파일의 끝에서 탈퇴한다. 기타 마지막이 아닌 모든 파일에 대해서는 파일별로 파일끝이라는것을 지적하는 대화식통보문을 내보낸다. 규정된 마지막파일에 대하여 또는 아무런 파일도 규정되지 않은 표준입력장치에 대하여 more 는 파일의 끝을 지적하는 대화식통보문을 내보내고 보충적인 지령들을 받아 들인다. 다음번 지령이 전진흐름을 규정하면 more 는 탈퇴한다. -e 가 규정되는 경우에 more 는 마지막파일의 마지막행을 쓴후에 탈퇴한다.

more 는 환경변수 MORE 를 리용하여 필요한 기발들을 설정한다. 임의의 지령행기발이나 인수들이 다음의 지령에서와 같이 MORE 변수뒤에서 처리된다.

more $ MORE flags arguments

실례로 -c 방식의 연산으로 파일을 보기 위해서는 다음과 같이 한다.

MORE='-c'; 외부적인 MORE 혹은 csh 지령

setenv MORE –c11

이 지령은 man 이나 msgs 와 같은 프로그람에 의한 주문도 포함하여 MORE 의 모든 주문내용들이 이 방식을 리용하도록 한다. MORE 환경변수를 설정하는 지령은 보통 **.**profile 혹은 **.**chrcs 파일에 놓인다.

현재의 위치는 다음과 같은 두가지 문제와 관련된다.

- 화면에서 현재행의 위치
- 화면에서 현재행의 행번호(파일안에서의 행번호)

현재위치에 대응하는 화면의 행은 화면우에서 세번째 행이다. 만일 이것이 불가능(현시하는 행이 3 보다 작거나 파일의 처음 혹은 마지막페지에 있는 경우)하다면 현재위치는 화면의 첫행 또는 마지막행이다.

다음의 령은 more 가 림시정지했을 때 입력될수 있는것들이다(여기서 i 는 옹근수인수로서 선택적으로 규정되며 기정값은 1 이다.).

i <Enter>
ij
i <Ctrl-e>
i <Space>

> i 개 행만큼 전진방향으로 흘러 보낸다. <Space>에 대한 기정값은 한개 화면이고 j 및 <Enter>에 대해서는 한개 행이다. i 가 화면크기보다 크다고 하여도 전체 i 개 행이 씌여 진다. 파일의 끝에서 <Enter>는 more 가 다음 파일을 계속하도록 하며 현재파일이 마지막파일이면 탈퇴한다.

id
i <Ctrl-d>

> i 개 행만큼 전진방향으로 흘러 보내는데 화면의 절반크기값을 기정으로 보낸다. i 가 규정되면 그 값이 d 및 u 지령에 대한 새로운 기정값으로 된다.

iu
i <Ctrl-u>

> i 개 행만큼 후진방향으로 흘러 보내는데 화면의 절반크기값을 기정으로 보낸다. i 가 규정되면 그 값이 d 및 u 지령에 대한 새로운 기정값으로 된다.

ik
i <Ctrl-y>

> 한 행을 기정값으로 가지면서 화면을 후진방향으로 흘러 보낸다. i 가 화면크기보다 크다고 하여도 전체 i 개 행이 현시된다.

iz

> i 개의 행들을 현시하며 창문의 크기를 새롭게 i 로 설정한다.

ig

1을 기정값으로 가지면서 파일의 i번째 행으로 이행한다. 그리고 그 행이 현재위치가 되도록 화면을 다시 현시한다. i가 규정되지 않으면 파일에서 첫 화면부분을 현시한다.

iG

파일의 끝을 기정값으로 가지면서 파일의 i번째 행으로 이행한다. i가 규정되지 않으면 파일의 마지막행이 화면의 밑에 놓이도록 화면을 다시 현시한다. i가 규정되면 현재행이 i번째 행으로 되도록 한다.

is

1을 기정값으로 가지며 i개 행만큼 전진방향으로 뛰여 넘고 그 점에서부터 시작하여 한개 화면분량을 현시한다. 만일 i에 의한 현재위치가 현시하여야 할 한개 화면분량보다 작게 되면 파일의 마지막화면분량이 현시된다.

if

i <Ctrl-f>

한개 화면분량을 기정값으로 가지면서 i개 행만큼 전진방향으로 뛰여 넘는다. 파일의 끝에서 more는 다음파일을 계속하며 현재파일이 마지막파일이면 탈퇴한다.

ib

i <Ctrl-b>

한개 화면분량을 기정값으로 가지면서 i개 행만큼 후진방향으로 뛰여 넘는다. i가 화면의 크기보다 작으면 마지막화면분량만이 현시된다.

q, Q, **:**q, **:**Q, zz

more로부터 탈퇴한다.

=

:f

<Ctrl-g>

현재 검사되고 있는 파일의 이름, 총 파일개수에 대한 상대적인 번호, 현재 행번호, 현재 바이트번호, 총 바이트수, 현재위치의 퍼센트를 현시한다. 이런 항목들은 씌여 진 행의 첫 바이트를 참조하여 얻어 진다.

v

현재 검사되는 파일을 편집하기 위한 편집기를 끌어 들인다. 편집기의 이름은 EDITOR 환경변수로 주어 진다. 기정은 vi

이다. 만일 EDITOR 가 vi 혹은 ex 를 가리킨다면 편집기는 현재의 편집기의 행이 more 지령의 현재위치에 대응하는 물리적행으로 되도록 입장한다.

편집기가 탈퇴할 때 more 는 현재위치에서 화면을 다시 현시하여 원래 상태를 귀환한다.

h

more 지령전반에 대한 해설을 현시한다.

i/[!] expression

주어 진 식 expression 을 포함하는 행들중에서 i 번째 행을 파일의 전진방향으로 탐색한다. i 의 기정값은 1 이다. 탐색은 현재위치의 다음행으로부터 출발한다. 탐색이 성과적으로 진행되면 화면은 탐색된 행이 현재위치에 놓이도록 수정된다. 빈 식(/<Enter>)은 이전 식에 대한 탐색을 다시 한다. !가 포함되여 있으면 주어 진 식을 포함하지 않는 행들을 탐색한다.

식의 발생개수가 i 보다 작고 입력파일이 파이프가 아니라면 파일안에서 그 위치는 변경되지 않은채로 남아 있다.

i?[!] expression

파일에서 후진방향으로 탐색한다는것을 제외하고는 우의것과 동일하다.

in

마지막으로 규정한 식을 포함하는 i 번째 행에 대한 이전탐색을 다시 한다(이전 탐색이 /! 혹은 ?!였다면 마지막식을 포함하지 않는 이전탐색을 다시 한다.). i 의 기정값은 1 이다.

iN

마지막식을 포함하는 i 번째 행에 대한 이전탐색의 반대방향으로의 탐색을 진행한다. i 의 기정값은 1 이다.

' '

제일 마지막의 큰 이행명령문이 수행되였던 위치로 귀환한다(한개 화면분을 넘어나는 이행으로서 정의되는것이다.). 이러한 이행이 없었다면 파일의 시작으로 귀환한다.

!command

command 로서 쉘을 입장시킨다. command 에 있는 % 및 !기호들은 현재파일의 이름과 이전의 쉘지령으로 각각 교체된다. 만일 현재파일이름이 없다면 %는 전개되지 않는다. \% 및 \! 은 % 및 !로 교체된다.

:e[file]

E[file]

새 파일을 검사한다. file 인수가 규정되지 않으면 현재파일이 다시 검사된다. file 이 #(수자기호)문자이면 이전에 검사된 파일이 다시 검사된다.

i:n

다음파일을 검사한다. i 가 규정되면 지령행에서 규정된 i 번째 다음파일을 현시한다.

i:p

이전 파일을 검사한다. i 가 규정되면 지령행에서 규정된 i 번째 이전 파일을 현시한다.

it tagstrng

규정된 tagstring 으로 이행하며 화면을 다시 표시한다. 다음에 파일을 검사한다. i 가 규정되면 지령행에서 규정된 i 번째 다음파일을 현시한다.

m letter

현재행을 규정된 글자로 표식한다. 표식글자는 한개의 소문자로 되여야 한다.

' letter

규정된 글자에 의하여 이미 표식 붙은 위치로 귀환한다.

r

<Ctrl-l>

화면을 갱신한다.

R

입력완충기를 지우면서 화면을 갱신한다.

지령들은 즉시에 효과를 내게 된다. 즉 <Enter>건을 누를 필요가 없다. 지령기호가 주어 지면 곧 행지우기기호를 리용하여 수값인수들이 형성되는것을 무시할 수 있다.

표준출력장치가 대역적인 장치가 아니라면 more 는 cat(1)과 동등하다.

more 는 SIGWINCH 부호를 보장하며 창문크기가 변하는데 따라 화면을 다시 현시한다.

외부적영향

환경변수들

COLUMNS	체계가 선택한 수평화면크기를 무시한다.
EDITOR	v 지령을 사용하여 편집기를 선택하게 한다.
LANG	LANG 은 설정되지 않았거나 비여 있는 국제화변수들에 대한 기정값을 제공한다. LANG 이 설정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).
LC_ALL	이 변수가 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모든 값을 무시한다.
LC_CTYPE	본문의 단일/다중바이트기호들, 인쇄할수 있는 기호들의 분류, 정규식에서 기호모임식들에 대한 해석을 결정한다.
LC_MESSAGES	표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용과 표준적인 출력장치에 씌여 지는 비형식적인 통보의 지역을 결정한다.
NLSPATH	LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.
LINES	체계가 설정한 수직화면크기를 무시하고 한개 화면의 행개수를 리용한다. -n 은 LINES 변수보다 우선권을 가진다.
MORE	지령행에서 <-> 및 <Space>기호앞에 놓이는 선택항목들을 포함하는 기호렬을 결정한다. 임의의 지령행선택항목들은 MORE 변수안의 이런 기호렬보다 앞에 놓인다. MORE 변수는 화면의 행개수를 결정하는 TERM 및 LINES 보다 높은 우선권을 가진다.
TERM	말단장치의 종류이름을 결정한다.

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

하나의 파일을 보려면 다음과 같이 하여야 한다.

```
more  filename
```

nroff 출력을 미리 보려면 다음과 같이 하여야 한다.

```
nroff  -mm  +2  doc.n  |  more  -s
```

만일 파일이 표들을 포함한다면 다음과 같이 하여야 한다.

tbl file | nroff -mm | call |more -s

15개의 행으로 된 창문에서 stuff파일을 현시하고 여러개의 빈 행들을 하나의 빈 행으로 바꾸려면 다음과 같이 하여야 한다.

more -s -n 15 stuff

매개 파일의 마지막화면분량을 검사하려면 다음과 같이 하여야 한다.

more -p G file1 file2

현재위치에서 100개행을 이행하여 매 파일을 보려면 다음과 같이 하여야 한다(3번째 행 즉 98번행이 첫행으로 된다.).

more -p 100g file1 file2

현재위치에서 30개 행을 이행하여 tag를 포함하는 파일을 보려면 다음과 같이 하여야 한다.

more -t tag -p 30g

경고

표준오유장치 즉 파일서술자 2가 대화과정에 표준적으로 입력장치로 리용되는데 이 장치는 방향을 바꾸지 말아야 한다.

파일

usr/share/lib/terminfo/?/*   콤파일된 말단성능자료기지

저자

more는 마크 뉴들리맨, 캘리포니아종합대학, 버클리, OSF, HP에서 개발하였다.

관련항목

csh(1), man(1), pg(1), sh(1), term(4), terminfo(4), environ(5), lang(5), regexp(5)

표준일치

more: XPG4

pg

pg – 파일의 전부 혹은 부분을 폐지단위로 현시하는 려과지령이다.

pg(1)

이름

pg – 프로그람으로 복사된 말단들에 대한 파일읽기려과기이다.

형식

pg [-number] [-p string] [-cefns] [+linenumber] [+/pattern] [file …]

주의

pg와 more는 둘 다 류사한 경우에 사용된다(more(1)을 참고). pg에서는 more에서와 달리 본문의 강조표시기능이 제공되지 않는다. 그러나 지령 pg는 more에 의하여 제공되지 않는 여러가지 유용한 기능들을 가지고 있다.

해설

pg는 프로그람적으로 복사된 말단상에서 본문파일을 한번에 한개 화면씩 검사하도록 하는 본문파일려과기이다. 만일 -가 파일인수로 쓰이였거나 pg 지령행에 null 인수가 있으면 표준입력장치가 리용된다. 매개 화면은 통보문에 의하여 넘어 가며 새 폐지를 현시하기 위해서는 <Enter>건을 누른다. 다른 방법들은 아래에서 설명한다.

more는 이미 지나간 화면을 다시 보기 위해 뒤로 되돌아 갈수 있지만 pg는 그렇지 않다. 이것에 대한것은 아래에서 설명한다.
말단의 속성을 결정하는 지령에서 pg는 환경변수 TERM에 의하여 규정되는 말단형태에 대한 terminfo 자료를 읽어 들인다(terminfo(1)을 참고). 환경변수 TERM이 정의되여 있지 않으면 말단의 형태는 dumb으로 설정된다.

선택항목들

-number 이 선택항목은 pg지령이 표준적으로 사용되지 않고 전용화되였을 때 창문의 크기(행개수)를 규정하는 옹근수이다(행개수가 24개인 말단에서 기정창문의 크기는 23이다.).

-p string

이 선택항목이 설정되면 pg는 대화식통보문으로 이 문자렬을 사용한다. 만일 대화식통보문자렬에 %d가 있으면 %d 앞에 있는 문자렬은 현재 대화식통보문이 나온 폐지의 폐지번호로 바뀐다. 대화식통보문의 기정문자렬은 두점(:)이다.

-c 새 페지를 현시하기전에 화면을 지우고 **유표**(Cursor)를 화면의 첫 시작위치에 가져 다 놓는다. 말단형태에 대한 terminfo 자료에서 clear_screen 이 정의되지 않았으면 이 선택항목은 무시된다.

-e 이 선택항목이 규정되면 pg 는 매 파일의 끝에서 멎지 않는다.

-f 표준적으로 pg 에 의하여 화면의 크기보다 더 긴 행들은 잘리우는데 그러면 현시되는 본문들중에서 어떤 문자렬들은 이지러져 나오는 경우가 있다. -f 는 자르는 행들에서 pg 지령의 이 기능이 적용될수 없게 한다.

-n 표준적으로 지령들은 새 행문자로 끝나야 한다. 그런데 이 선택항목을 사용하면 지령글자들이 입력될 때 곧 행끝기호가 자동적으로 삽입된다.

-s 이 선택항목은 설정되면 pg 는 모든 통보문들을 두드러진 방식(보통 반전비데오)으로 인쇄한다.

+linenumber
이 행에서 현시를 시작한다.

+/pattern/
주어 진 패턴을 포함하는 첫행에서 시작한다.

pg 는 환경변수 PG 에서 필요한 기발들이 설정된것처럼 볼수 있다. 실례로 사용자가 -c 방식 즉 Bourne-쉘지령 PG='-c'를 리용하여 파일을 보려고 한다면 외부의 PG 혹은 C-쉘지령 setenve PG -c 는 man 과 msgs 와 같은 프로그람을 통한 주문까지 포함하여 pg 의 모든 주문내용들이 이 방식을 리용하도록 수행시킨다. PG 환경변수를 설정하기 위한 지령은 표준적으로 **.**profile 혹은 **.**cshrc 파일에 놓인다.

pg 가 림시정지했을 때 입력할수 있는 대답에는 세가지 종류가 있다. 즉 그것들은 계속읽기, 탐색하기, 읽기환경의 수정이다.

계속읽기(further perusal)는 표준적으로 앞주소를 취한다. 앞주소는 앞의 본문이 현시될 위치를 가리키는 부호가 붙을수 있는 주소이다. 이 주소는 지령에 따라 페지 혹은 행단위로 해석된다. 부호 있는 주소는 현재페지나 행에 대한 상대적인 위치를 규정하며 부호 없는 주소는 파일의 시작에 대한 상대적인 주소를 규정한다. 매개 지령은 아무런 주소도 제공되지 않았을 때 기정주소를 가진다.

읽기지령과 그 기정값들은 다음과 같다.

(+1) <newline> or <blank>
한개 페지를 현시한다. 그 주소는 페지별로 규정된다.

(+1) l
pg 는 상대주소를 가지고 화면을 전진 혹은 후진시키거나 규정된 행개수만큼 흘러 보낸다. 절대주소로 지적되면 pg 는 규정된 행부터 시작하여 한개 화면을 현시한다.

(+1) d 혹은 ^D
화면을 절반씩 전진 혹은 후진시킨다.

다음의 읽기지령들은 주소를 가지지 않는다.

. 혹은 ^L
한개 점을 입력하여 현재페지의 본문이 다시 현시되도록 한다.

$ 파일의 마지막창문부분을 현시한다. 입력이 파이프일 때 사용된다.

다음의 지령들은 본문에서 본문패턴에 대한 탐색을 진행할 때 쓰인다. 기본정규식문장론이 보장된다(regexp(5)를 참고). -n 이 규정되였다고 하여도 정규식은 언제나 새 행문자로 끝나야 한다.

i/pattern/
i 번째(기정값은 i=1)로 나타나는 패턴을 전진방향으로 탐색한다. 이 탐색은 현재페지의 가운데서부터 시작하며 현재파일의 끝까지 계속된다.

i^pattern^
i?pattern?
i 번째(기정값은 i=1)로 나타나는 패턴을 후진방향으로 탐색한다. 이 탐색은 현재페지의 가운데서부터 시작하며 현재파일의 시작까지 계속된다. ^표시는 ?를 조종할수 없는 100 개의 말단을 추가할 때 효과적이다.

탐색후에 pg 는 찾은 행을 화면의 꼭대기에 현시한다. 이 기능은 탐색지령에 m 혹은 b 를 추가하여 찾은 행을 화면의 중간 또는 밑에 현시하도록 변경할수 있다.

앞붙이 t를 붙여 초기의 위치를 재기억시킬수 있다. pg는 다음의 지령들로써 읽기환경을 수정할수 있다.

in　　　지령행에서 i번째의 다음파일을 읽기 시작한다. i는 부호 없는 수로서 기정값은 1이다.

ip　　　지령행에서 i번째의 이전파일을 읽기 시작한다. i는 부호 없는 수로서 기정값은 1이다.

iw　　　또 다른 본문창문을 현시한다. 창문의 크기를 i로 설정한다.

s filename

규정된 파일을 보관한다. 현재 읽고 있는 파일만이 보관된다. s와 filename 사이의 공백은 없을수도 있다. ?n이 규정되였다고 하여도 이 지령은 언제나 새 행문자로 끝나야 한다.

h　　　가능한 지령들에 대한 도움말을 현시한다.

q 혹은 Q

pg를 끝낸다.

!command

지령은 환경변수 SHELL로 규정된 쉘에로 통과된다. 만일 그것이 불가능하면 기정쉘이 리용된다. ?n이 규정되였다고 하여도 이 지령은 언제나 새 행문자로 끝나야 한다.

pg가 말단장치로 출력할 때 임의의 순간에 그 출력을 정지시키고 통보문을 현시하려면 탈퇴건(보통 Ctrl-\) 혹은 중단건(Break)을 누른다. 그다음 우의 지령들을 표준방식으로 입력시킬수 있다. 때때로 어떤 문자렬들이 말단의 출력대기렬에서 기다리거나 말단출력대기렬이 없어 진것으로 하여 일부 출력내용을 잃어 버릴수도 있다.

만일 표준출력장치가 말단이 아니라면 pg는 매개 파일앞에 머리부가 출력된다는것을 제외하고는 기능적으로 cat와 동일하다.

외부적영향

환경변수들

LC_COLLATE는 정규식을 판정하는데 리용되는 조사렬을 결정한다.

LC_CTYPE는 본문의 단일/다중바이트기호, 정규식에서 기호모임식에 의하여 맞추어 지는 기호들에 대한 해석을 결정한다.

LANG은 통보문이 현시되는 언어를 결정한다.

만일 LC_COLATE 나 LC_CTYPE 가 규정되지 않았거나 빈 기호렬로 설정되였으면 LANG 이 기정값으로 리용된다. 만일 LANG 이 결정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

체계가 news를 읽을 때 pg를 리용하려면 다음과 같이 하여야 한다.

news | pg -p "(Page %d):"

경고

만일 말단태브가 매 8 번 위치마다에 설정되여 있지 않으면 예견하지 않았던 결과가 발생할수 있다.

pg 를 crypt(1)과 같은 말단 I/O 를 변경시키는 다른 지령의 려과기로서 리용할 때 말단설정은 정확히 기억되지 않을수도 있다.

pg 가 말단으로부터 입력을 기다릴 때 사용자는 BREAK, DEL, ^으로 대답할수 있다. 그러나 이러한 기호들은 pg 의 현재과제를 중단시키며 사용자를 대기상태에 놓는다. 이것들은 파이프를 통하여 읽을 때 리용되여야 한다. 왜냐하면 이러한 중단이 파이프행에 있는 다른 지령들도 정지시킬수 있기때문이다.

z 및 f지령들도 리용할수 있으나 /, ^, ?기호들은 패턴탐색지령에서 생략될수 있다.

파일

/usr/share/lib/terminfo/?/* 말단정보자료기지

/tmp/pg* 입력자료가 파이프로부터 올 때 리용하는 림시파일

관련항목

crypt(1), grep(1), more(1), terminfo(4), environ(5), lang(5), regexp(5)

표준일치

pg: SVID2, SVID3, XPG2, XPG3

tail

tail - 파일의 마지막부분을 현시한다.

tail(1)

이름

tail - 파일의 마지막부분을 현시한다.

형식

tail [-f] [b number] [file]
tail [-f] [c number] [file]
tail [-f] [n number] [file]

절대형식:

tail [+/-] [number] [l | b | c] [-f] [file]

해설

tail 은 주어 진 파일을 해당한 장소에서 시작하여 표준출력장치에 복사한다. 파일이 규정되지 않았으면 표준입력장치를 리용한다.

지령형식들

tail -b number …

파일을 그것의 끝 혹은 시작으로부터 number 만한 크기의 블로크만큼 복사한다.

tail -c number …

파일을 그것의 끝 혹은 시작으로부터 number 만한 크기의 바이트만큼 복사한다.

tail -n number … 혹은 tail number

파일을 그것의 끝 혹은 시작으로부터 number 만한 크기의 행만큼 복사한다.

tail 이 아무런 선택항목도 가지지 않으면 tail -n 10 과 동일하다.

선택항목 및 지령행인수들

-f 흐름선택항목이다. 입력파일이 표준파일이거나 FIFO 를 규정하면 입력

파일이 다 복사된후에 끝내지 않고 앞의 바이트들을 읽고 복사한다(tail 은 입력파일로부터 두번째로 읽고 복사하기 위하여 무한순환을 하게 된다.). 이것은 본문을 쓰기조종하면서 읽을 때 효과적이다. 인수 file 이 규정되지 않았으면 입력은 파이프(FIFO)이며 -f 는 무시된다.

number 복사되는 출력량의 단위를 규정하는 10 진수이다. number 앞에 +부호가 있으면 복사는 파일의 시작부터 number 단위만큼 내려 와서 진행되며 -부호가 있으면 복사는 파일의 끝에서부터 number 단위만큼 올라 가서 진행된다. number 앞에 b, c, n 이 없으면 ?n 으로 가정된다. 이러한 선택항목 혹은 number 가 둘 다 규정되지 않으면 -n 10 으로 가정된다.

-b number

파일의 끝 혹은 시작부터 number 만한 512byte 크기의 블로크만큼 환산하여 파일을 복사한다. number 가 규정되지 않으면 -b 10 으로 가정된다.

-c number

파일의 끝 혹은 시작부터 number 만한 바이트만큼 환산하여 파일을 복사한다. number 가 규정되지 않으면 -c 10 으로 가정된다.

-n number

파일의 끝 혹은 시작부터 number 만한 행만큼 환산하여 파일을 복사한다. number 가 규정되지 않으면 -n 10 으로 가정된다.

file 복사되는 파일의 이름이다. 규정하지 않으면 표준입력장치가 리용된다.

-c 가 규정되면 입력파일은 임의의 자료를 포함할수 있다. 그밖의 경우에는 입력파일이 본문파일이여야 한다.

절대형식에서 선택항목문자들은 블로크, 바이트 혹은 행들을 선택하는 number 인수뒤에 놓여야 한다. number 가 규정되지 않으면 -10 으로 가정된다.

외부적영향

환경변수들

LC_CTYPE 는 본문의 단일/다중바이트기호에 대한 해석을 결정한다.
LC_MESSAGES 는 통보문이 현시되는 언어를 결정한다.
만일 LC_CTYPE 혹은 LC_MESSAGES 가 규정되지 않았거나 빈 기호렬로 설정되였으면 LANG 이 기정값으로 리용된다. 만일 LANG 이 결정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 tail 은 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다. 그러나 선택항목 b 및 c 들은 다중바이트기호들을 자를수 있으므로 따라서 그 기호들은 다중바이트국부변수에서 리용되여야 한다.

실례

file1 의 마지막 3 개 행을 표준출력장치로 현시하고 tail 이 흐름방식으로 남아 있도록 하려면 다음과 같이 하여야 한다.

```
tail -f n 3 file1
```

혹은

```
tail -3 -f file1
```

logfile 의 마지막 15 개 행을 현시하고 tail 이 사명을 다할 때까지 logfile 의 앞으로 계속 나가도록 하려면 다음과 같이 하여야 한다.

```
tail -f c 15 logfile
```

혹은

```
tail -f -c 15 logfile
```

전체 파일을 현시하는 방법에는 다음과 같은 세가지가 있다.

```
tail -b +1 file
tail -c +1 file
tail -n +1 file
```

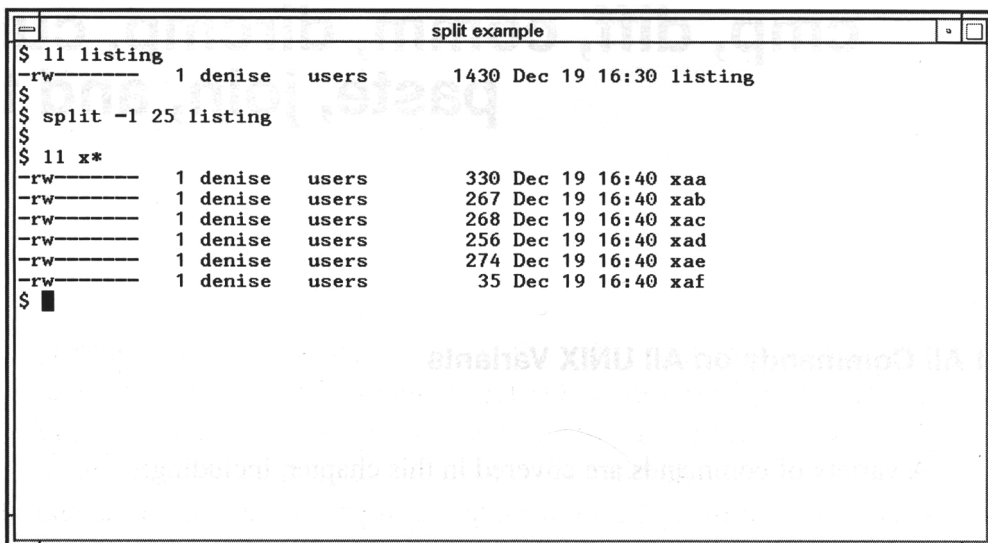
경고

파일의 끝에서부터 20Kbyte 의 꼬리부분이 완충기에 기억된다. 따라서 다른 지령으로부터의 결과를 tail 에 파이프로 련결시킬 때 주의하여야 한다.

기호파일들에 대하여 여러가지 종류의 레외적인 현상이 발생될수 있다.

관련항목

dd(1), head(1)

표준일치

tail: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제5장. UNIX 도구 - split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, tr 지령

이 장에서는 UNIX 의 기본도구로 되는 split, wc, sort, cmp, diff, comm, dircmp, cut, paste, join, tr 지령들에 대하여 고찰한다. 이 지령들은 대단히 효과적이며 즐겨 사용할수 있는것들이다. 그러나 모든 UNIX 변종들에서 이런 지령들이 다 가능한것은 아니다. 자기의 체계에 목적하는 지령이 없으면 류사한 지령을 사용하거나 여러개의 지령들을 결합하여 목적하는 작업을 할수 있다.

split

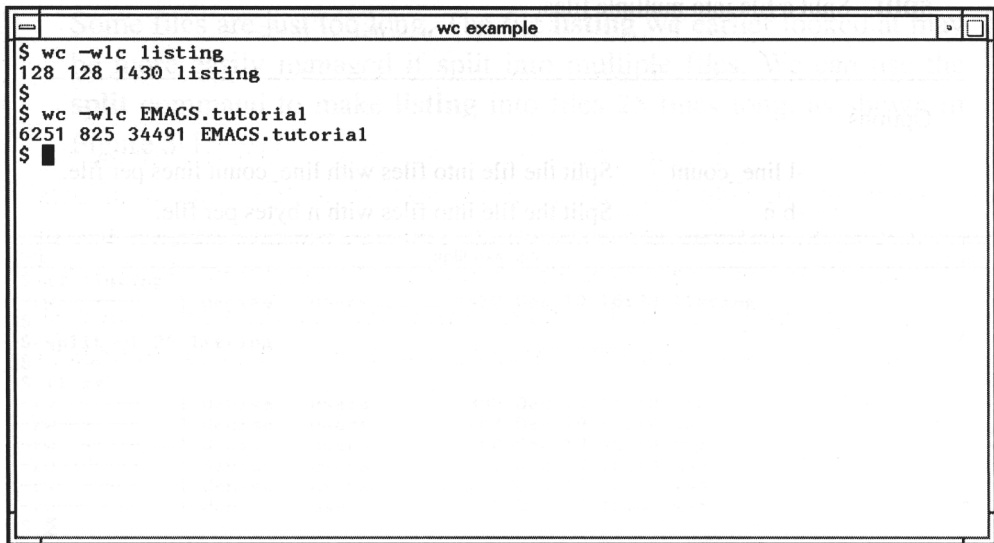
일부 파일들은 대단히 길기때문에 split 지령을 사용하여 여러개의 파일들로 갈라서 쉽게 관리할수 있다. 그림 5-1 에서처럼 split 지령에 의하여 listing 파일을 25 개의 행으로 된 파일들로 가를수 있다.

```
split example
$ ll listing
-rw-------    1 denise   users       1430 Dec 19 16:30 listing
$
$ split -l 25 listing
$
$ ll x*
-rw-------    1 denise   users        330 Dec 19 16:40 xaa
-rw-------    1 denise   users        267 Dec 19 16:40 xab
-rw-------    1 denise   users        268 Dec 19 16:40 xac
-rw-------    1 denise   users        256 Dec 19 16:40 xad
-rw-------    1 denise   users        274 Dec 19 16:40 xae
-rw-------    1 denise   users         35 Dec 19 16:40 xaf
$ █
```

그림 5-1. 지령 split

split 지령은 listing 파일을 xaa, xab 등의 여러개 파일들로 나누었다. split 에서 -l 은 생성되는 파일의 행개수를 규정한다. 다음의 내용은 split지령에 대한 개요이다.

split - 하나의 파일을 여러개 파일들로 가른다.

선택항목들

-l line_count

파일을 line_count 만한 행개수를 가지는 파일들로 가른다.

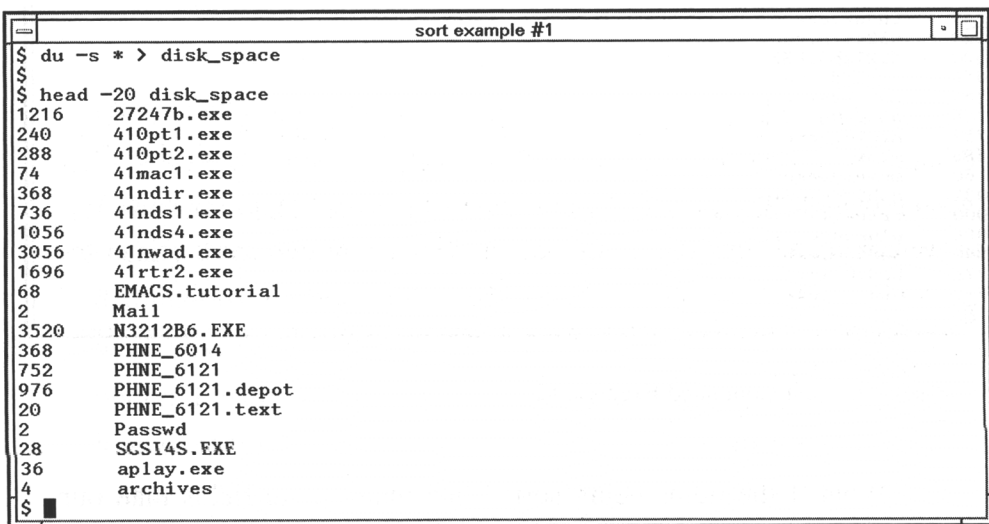
-b n 파일을 n 바이트크기의 파일들로 가른다.

wc

앞에서 listing 파일을 25 개 행을 가지는 여러개의 파일로 분리하였지만 listing 에 초기에 몇개의 행이 있었는지, 몇개의 단어가 있었는지 또한 몇개의 문자들이 있었는지는 알수 없다. wc 지령은 단어, 행, 문자개수를 계산하는 지령이다. 그림 5-2 에서 보여 준것처럼 -w 선택항목은 단어의 개수, -l 은 행개수, -c 는 문자개수를 계산한다.

그림 5-2. 지령 wc

listing 파일의 단어개수와 행개수가 동일하게 생성된것은 매행이 꼭 하나의 단어만을 포함한다는것을 의미한다. 우의 그림에서 EMACS.tutorial 이라는 본문파일에 작용된 wc 지령은 단어개수 6251, 행개수 825, 기호개수 34491 이라는 결과를 내보냈다. 즉 여기에서는 행개수보다 단어개수가 많은것이다. 다음의 내용은 wc 지령에 대한 개요이다.

wc – 단어, 행, 기호들의 개수를 생성한다.

선택항목들

-l 행개수를 인쇄한다.
-w 단어개수를 인쇄한다.
-c 기호개수를 인쇄한다.

sort

때때로 파일의 내용은 자기가 요구하는대로 정돈되여 있지 않을수도 있다. sort 지령을 사용하여 여러가지 방법으로 파일을 정돈할수 있다.

UNIX 체계를 리용할 때 체계관리자는 사용자가 소비하는 디스크공간의 량을 관리한다. 사용자는 du 지령으로 자기가 소비하는 디스크공간의 량을 조절할수 있다. 그림 5–3에서는 disk_space 라는 파일을 창조하고 파일과 등록부에 의하여 소비된 디스크의 량을 보여 주고 있다. 그리고 그 파일의 처음 20개 행도 현시한다.

```
sort example #1
$ du -s * > disk_space
$
$ head -20 disk_space
1216    27247b.exe
240     410pt1.exe
288     410pt2.exe
74      41mac1.exe
368     41ndir.exe
736     41nds1.exe
1056    41nds4.exe
3056    41nwad.exe
1696    41rtr2.exe
68      EMACS.tutorial
2       Mail
3520    N3212B6.EXE
368     PHNE_6014
752     PHNE_6121
976     PHNE_6121.depot
20      PHNE_6121.text
2       Passwd
28      SCSI4S.EXE
36      aplay.exe
4       archives
$ █
```

그림 5–3. 지령 sort의 실례 #1

결과는 보통 자모순서로 정돈되는것이 정상이다. 그림 5–4에서는 sort 지령을 리용하여 자모순서대로가 아니라 수값크기순서 즉 파일 및 등록부들이 소비한 디스크공간이 큰것부터 순서대로 출력된다. sort 지령의 -n 은 수값크기순서로 결과가 정돈되게 하고 -r 는 결과가 반대의 수값크기순서로 정돈되게 하며 -o 는 출력파일의 이름을 결정한다.

```
sort example #2
$ sort -n -r disk_space -o disk_space_numeric
$
$ head -20 disk_space_numeric
288238  main.directory
60336   emacs-19.28.tar
8128    c3295n_a.exe
5024    trace.TRC1
4496    rkhelp.exe
3840    tnds2.exe
3520    n32e12n.exe
3520    N3212B6.EXE
3056    41nwad.exe
2784    nsh220e2.zip
2768    nsh220e3.zip
2752    nfs197.exe
2688    mbox
2160    msie10.exe
2032    nsh220e1.zip
2000    trace.TRC1.Z.uue
1984    plusdemo.exe
1840    ja95up.exe
1776    hpd10117.exe
1712    wsos22.exe
$
```

그림 5-4. 지령 sort의 실례 #2

정돈될 항목들이 두개이상의 마당으로 되는 경우에는 보다 복잡한 정돈을 해야 한다. 실례로 passwd.test라는 파일을 다시 표시하면 다음과 같다.

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:WhOyzfAV2qm2Y:100:100:Caldera OpenLinux
                                   User:/home/col:/bin/bash
```

먼저 어느 사용자들이 같은 그룹에 속하는가를 결정해 보자. passwd.test 파일에서 매 마당들은 두점(:)으로 분리되여 있다. 여기서 네번째 마당은 사용자가 속하는 그룹을 표시한다. 실례에서 bin 은 그룹 1 에 속하며 daemon 은 그룹 2 에 속한다. 그룹별로 정돈하자면 sort 지령에서 세개의 선택항목들을 규정하여야 한다. 첫 선택항목은 -t 로서 마당분리기호 두점(:)을 규정하는것이다. 그다음 선택항목은 -k 로서 정돈해야 할 마당을 규정하는것이다. 마지막선택항목은 -n 으로서 수값정돈을 규정한다. 다음의 실례는 passwd.test 파일의 네번째 마당을 수값순서로 정돈한 결과를 보여 주고 있다.

```
# sort -t: -k4 -n passwd.test
halt:*:7:0:halt:/sbin:/sbin/halt
operator:*:11:0:operator:/root:
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
sync:*:5:0:sync:/sbin:/bin/sync
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
man:*:15:15:Manuals Owner:/:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
col:WhOyzfAV2qm2Y:100:100:Caldera OpenLinux
                    User:/home/col:/bin/bash
games:*:12:100:games:/usr/games:
nobody:*:65534:65534:Nobody:/:/bin/false
```

아래에서는 지령 sort 의 개요를 보여 주고 있다.

sort – 파일의 행들을 정돈한다.(기정값은 자모순정돈이다.)

선택항목들

-b	공백과 태브를 무시한다.
-c	파일이 이미 정돈되였는가를 검사하고 정돈되여 있으면 아무것도 하지 않는다.
-d	등록부순서로 정돈한다.
-f	정돈할 때 대소문자를 무시한다.
-i	정돈할 때 비 ASCII 기호를 무시한다.
-ks	정돈해야 할 마당을 s 로 한다.

-m 정돈된 파일들을 결합한다.
-n 수값순서로 정돈한다.
-o file 출력파일을 표준출력장치가 아니라 file로 규정한다.
-r 반대순서로 정돈한다.
+n 정돈하기전에 n개의 마당 혹은 렬들을 뛰여 넘는다.

cmp, diff, comm

일상생활에서는 파일들을 많이 편집하는데 사용자는 때때로 어느 파일에서 무엇을 편집하였는지 잊어 버릴수도 있다. 이때는 파일들을 비교해야 할 필요가 있을것이다. cmp, diff, comm은 파일들을 비교하는 지령이다.

llsum이라는 파일을 수정하였다고 하자. 수정하지 않았을 때 그것은 llsum.orig로 보관되여 있었다. head지령을 사용하여 llsum 및 llsum.orig의 첫 20개 행을 각각 현시하면 다음과 같다.

```
# head -20 llsum
#
# !/bin/sh
# Displays a truncated long listing ( l l ) and
# displays size statistics
# of the files in the listing.

l l $* | \
awk ' BEGIN { x=i=0; printf "%--25s%-10s9k8s%8s\n" , \
                    "FILENAME" , "OWNER'f, l'SIZE" , "TYPE"
          $ l ~ / ^ [ -dlps ] / { # line format for normal files
               printf "%-25s%-10s%8d" , $9 , $3 , $5
               x = x + $5
               i++
                    }
          $1 ~ / ^ - / { printf "%8s\n", "file" } # standard file types
          $1 ~ / ^ d/ { printf "%8s\n", "dir" }
          $1 ~ / ^ l/ { printf "%8s\n", "l~nk" }
          $1 ~ / ^ p/ { printf "%8s\n", "pipe" }
          $1 ~ / ^ s/ { printf "%8s\n", "socket" }
          $1 ~ / ^ [ bc ]/ { # line format for device files
                         printf "%-25s%-
10s%8s%8s\n", $10, $3, " ", "dev"
                         }
```

```
#
# head  -20  llsum.orig
#
# !/bin/sh
# Displays  a  truncated  long  listing  ( l l )  and
# displays  size  statistics
# of  the  files  in  the  listing.

l l $* | \
awk ' BEGIN { x = i = 0 ; printf "%-16s%-10s%-8s%8s\n",   \
                              "FILENAME",   "OMMER",   "SIZE",   "TYPE" }
            $1  ~ / ^ [ -dlps ]  {# line format for normal files
                              printf "%-16s%-10s%8d", $9, $3, $5
                              x = x + $5
                              i++
                              }
            $1  ~ / ^ - / {    printf "%8s\n",   "file"     # standard file
types
            $1  ~ / ^ d /       { printf     "%8s\n", "dir"}
            $1  ~ / ^ l /       { printf     "%8s\n", :link"}
            $1  ~ / ^ p /       { printf     "%8s\n", Ipipe"}
            $1  ~ / ^ s /       { printf H %8s\n", "socket"}
            $1  /^  [ bc ]   l/  { # line format for device files
                                     printf                    "%-25s%-
10s%8s%8s\n",   $10,   $3,   " ",   "dev"
```

llsum.orig 를 개선하기 위하여 무엇을 변경했는지 잘 모르면 cmp 지령으로 먼저 두 파일사이에 차이점이 실지로 존재하는가를 다음과 같이 알아 볼수 있다.

```
$
$ cmp  llsum  llsum.orig
llsum  llsum.orig  differ:  char  154,  line  6
$
```

cmp 는 두 파일에서 6 번째 행의 154 번째 기호가 차이난다는 간단한 정보만을 현시한다. 즉 cmp 지령으로는 두 파일에 차이가 있다는것만을 확인하고 그 차이가 얼마나 큰 것인가에 대해서는 알수 없다.

두 파일에서 차이나는 모든 정보를 얻으려면 -l 을 리용해야 한다.

```
$ cmp -l llsum llsum.orig
    154      62      61
    155      65      66
    306      62      61
    307      65      66
    675      62      61
    676      65      66
```

두 파일의 차이점위치만이 아니라 어떻게 차이나는가도 알고 싶다면 diff 지령을 사용한다.

```
$ diff llsum llsum.orig
6c6
< awk     ' BEGIN {x=i=0; printf "%-25s%-10s%8s%8s\n", \
…
> awk     ' BEGIN {x=i=0; printf "%-16s%-10s%8s%8s\n", \
9C9
<  printf "%-25s%-10s%8d",  $9,  $3,  $5
…
>  printf "%-16s%-10s%Bd",  $9,  $3,  $5
19C19
<  printf "%-25s%-10s%Bs%8s\n",  $10,  $3,  "",  "dev"
…
>  printf "%-16s%-10s%8s%8s\n",  $10,  $3,  "",  "dev"
$
```

이 실례에서 보여 준바와 같이 두개의 파일에서는 6, 9, 19 행들이 차이난다는것을 알수 있다. 즉 llsum.orig 에서 수 16 이 llsum 에서는 25 로 변했다는것과 이 부분이 두 파일에서 차이나는 점의 전부라는것을 알수 있다.

기호 "<"는 첫번째 파일(llsum)의 행앞에 놓이며 기호 ">"는 두번째 파일(llsum.orig)의 행앞에 놓인다. 사용자가 llsum 파일에서 변경한 내용은 두번째 그룹의 정보가 기호 16 부터가 아니라 기호 25 부터 시작되도록 한것이다. 즉 25 번째 렬부터 두번째 그룹정보가 생성되도록 하였다. 다음의 실례에서 보여 준바와 같이 두번째 그룹정보는 소유자이다.

```
$ llsum
FILENAME        OWNER     SIZE    TYPE
README          denise     810    file
backup_files    denise    3408    file
biography       denise     427    file
```

```
cshtest                 denise     1024    dir
gkill                   denise     1855    file
gkill.out               denise      191    file
hostck                  denise      924    file
ifstat                  denise     1422    file
ifstat.int              denise     2147    file
ifstat.out              denise      723    file
introdos                denise    54018    file
introux                 denise    52476    file
letter                  denise    23552    file
letter.auto             denise    69632    file
letter.auto.recover     denise    71680    file
letter.backup           denise    23552    file
letter.lck              denise       57    file
letter.recover          denise    69632    file
llsum                   denise     1267    file
llsum.orig              denise     1267    file
llsum.out               denise     1657    file
llsum.tomd.out          denise     1356    file
PS9                     denise      670    file
psg.int                 denise      802    file
psg.out                 denise      122    file
sam-adduser             denise     1010    file
tdolan                  denise     1024    dir
trash                   denise     4554    file
trash.out               denise      329    file
typescript              denise     2017    file

The files listed occupy 393605 bytes (0.3754 Mbytes)
Average file size is 13120 bytes
$
```

이제 llsum.orig 를 실행시키면 두번째 그룹정보 즉 소유자는 명백히 32 번째 렬이 아니라 16 번째 렬로부터 시작한다.

$ 11sum.orig

```
FILENAME          OWNER      SIZE    TYPE

README            denise      810    file
backup_files      denise     3408    file
biography         denise      427    file
```

cshtest	denise	1024	dir
gkill	denise	1855	file
gkill.out	denise	191	file
hostck	denise	924	file
ifstat	denise	1422	file
ifstat.int	denise	2147	file
ifstat.out	denise	723	file
introdos	denise	54018	file
introux	denise	52476	file
letter	denise	23552	file
letter.auto	denise	69632	file
letter.auto.rec	denise	71680	file
letter.backup	denise	23552	file
letter.lck	denise	57	file
letter.recover	denise	69632	file
llsum	denise	1267	file
llsum.orig	denise	1267	file
llsum.out	denise	1657	file
llsum.tomd.out	denise	1356	file
psg	denise	670	file
psg.int	denise	802	file
psg.out	denise	122	file
sam_adduser	denise	1010	file
tdolan	denise	1024	dir
trash	denise	4554	file
trash.out	denise	329	file
typescript	denise	3894	file

The files listed occupy 395482 bytes（0.3772 Mbytes）
Average file size is 13182 bytes

script done on Mon Dec 11 12:59:18
$

comm 지령을 리용하면 정돈된 파일들을 비교하여 매 파일에 고유한 행들을 볼수 있다. comm 지령으로 두 파일을 비교하면 첫 부분에는 첫번째 파일에만 있는 행들이 나타나고 두번째 부분에는 두번째 파일에만 있는 행들이 나타나며 세번째 부분에는 두 파일에 다 있는 행들이 현시된다. 다음의 실례에서는 두개의 /etc/passwd 파일들을 비교한 결과를 보여 주고 있다.

comm /etc/passwd /etc/pasawd.backup

root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash

```
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operatorl:*:12:0:operator:/root:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP  User:/home/ftp:
man:*:15:15:Manuals  Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:WhOyzfAV2qm2Y:100:100:Caldera
OpenLinux  User:/home/col:/bin/bash
```

이 실례에서 보여 주는것처럼 사용자 games 는 현재의 /etc/passwd 파일에만 있고 사용자 oprator1 은 낡은 /etc/passwd.backup 파일에만 있으며 기타 나머지는 두개의 파일에 다 있다.

아래에서는 지령 cmp 와 diff 에 대한 개요를 준다.

cmp - 두 파일의 내용을 비교한다. 처음으로 차이나는 행번호와 바이트위치를 귀환한다.

선택항목들

-l	한 파일안에서 차이나는 바이트위치와 차이나는 기호들을 모두 현시한다.
-s	탈퇴코드만 귀환한다.

diff - 두 파일을 비교하여 차이나는 행들을 출력한다.

선택항목들

-b	행끝에서 공백들을 무시한다.
-i	대소문자차이를 무시한다.
-t	태브들을 출력에서 공백으로 한다.
-w	공백과 태브들을 무시한다.

dircmp

dircmp는 두개의 등록부를 비교하고 등록부들의 내용에 대한 정보를 제시한다. 아래에서는 두개의 긴 등록부들에 대한 렬거내용을 보여 주고 있다.

```
$ ls -l krsort.dir.old
total 168
-rwxr-xr-x   1   denise   users   34592   Oct   31   11:27   krsort
-rwxr-xr-x   1   denise   users    3234   Oct   31   11:27   krsort.c
-rwxr-xr-x   1   denise   users   32756   Oct   31   11:27   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   31   11:27   krsort.q
-rwxr-xr-x   1   denise   users    3085   Oct   31   11:27   krsortorig.c
$
$ ls -l krsort.dir.new
total 168
-rwxr-xr-x   1   denise   users   34592   Oct   31   15:17   krsort
-rwxr-xr-x   1   denise   users   32756   Oct   31   15:17   krsort.dos
-rw-r--r--   1   denise   users    9922   Oct   31   15:17   krsort.q
-rwxr-xr-x   1   denise   users    3234   Oct   31   15:17   krsort.test.c
-rwxr-xr-x   1   denise   users    3085   Oct   31   15:17   krsortorig.c
$
```

우에서 알수 있는바와 같이 krsort.c는 krsort.dir.old 등록부에만 있으며 krsort.test.c는 krsort.dir.new 등록부에만 있다는것을 알수 있다.

이제 dircmp지령을 사용하여 이 차이점들에 대한 정보를 보기로 하자.

```
$ dircmp krsort.dir.old krsort.dir.new

krsort.dir.old only and krsort.dir.new only Page 1

./krsort.c    ./krsort.test.c

Comparison of krsort.dir.old krsort.dir.new Page 1

directory
same        ./krsort
same        ./krsort.dos
same        ./krsort.q
same        .  /krsortorig.c
$
```

출력결과를 보면 먼저 한 등록부에만 있는 파일들이 각각 렬거된 다음에 두 등록부

에 공통적인 파일들이 렬거된다.

다음은 dircmp 지령에 대한 개요이다.

dircmp - 등록부들을 비교한다.

선택항목들

-d	두개의 등록부에서 같은 이름을 가지는 파일들의 내용을 비교하고 파일들이 동일하자면 무엇을 해야 하는가를 현시한다.
-s	차이나는 파일들에 대한 정보를 금지한다.

cut

지령 cut 를 리용하면 출력결과의 마당들을 잘라 몇개의 마당만 현시하게 할수 있다. 실례로 앞에서 고찰한 llsum 파일의 출력은 파일이름, 소유자, 파일크기, 파일종류라는 네개의 마당을 가지고 있었다. 다음의 실례에서는 지령 cut 를 리용하여 llsum 의 소유자와 파일종류마당을 자르고 나머지마당들만 현시하도록 하고 있다.

```
$ llsum | cut -C 1-25, 37-43
FILENAMESIZE
README                  810
backup_files           3408
biography               427
cshtest                1024
gkill                  1855
gkill.out               191
hostck                  924
ifstat                 1422
ifstat.int             2147
ifstat.out              723
introdos              54018
introux               52476
letter                23552
letter.auto           69632
letter.auto.recover   71680
letter.backup         23552
letter.lck               57
letter.recover        69632
llsum                  1267
llsum.orig             1267
llsum.out              1657
```

```
llsum.tomd.out           1356
psg                       670
psg.int                   802
psg.out                   122
sam_adduser              1010
tdolan                   1024
trash                    4554
trash.out                 329
typescript                 74

The files listed occupy 3 (0.373
Average file size is 1305
$
```

이 실례에서는 llsum을 파이프를 리용하여 cut에 전달한 결과를 보여 주고 있다. 즉 1번 기호로부터 25번까지의 기호들과 37번부터 43번까지의 기호들이 현시되였다. 그런데 출력의 마지막 두개 행에서 기호들이 잘리우고 일부만 현시되였다. 이런 현상을 피하기 위하여 grep -v 지령을 사용하여 기타 행들에서는 전부가 출력되게 할수 있다. 아래 실례의 마지막부분에서는 이 지령의 출력이 llsum.out파일에 보관되도록 하고 있다.

$ ./llsum | grep -v "bytes" | cut -c 1-25, 37-43

```
FILENAME             SIZE
README                810
backup_files         3408
biography             427
cshtest              1024
gkill                1855
gkill.out             191
hostck                924
ifstat               1422
ifstat.int           2147
ifstat.out            723
introdos            54018
introux             52476
letter              23552
letter.auto         69632
letter.auto.recover 71680
letter.backup       23552
letter.lck             57
```

```
letter.recover          69632
llsum                    1267
llsum.orig               1267
llsum.out                1657
llsum.tomd.out           1356
psg                       670
psg.int                   802
psg.out                   122
sam_adduser              1010
tdolan                   1024
trash                    4554
trash.out                 329
typescript               1242
```

$ **llsum | grep -v "bytes" | cut -c 1-25, 37-43 > llsum.out**
$

아래에서는 cut지령에 대한 개요를 보여 주고 있다.

cut - 매개 행에서 규정된 마당들만 추려 낸다.

선택항목들

-c list	기호위치에 토대하여 추려 낸다.
-f list	마당에 토대하여 추려 낸다.
-d char	char는 -f에서 마당을 구별하는 분리기호이다.

paste

파일들은 여러가지 방법으로 결합할수 있다. 행단위로 파일들을 결합할 때에는 paste 지령을 사용할수 있다. paste 지령을 사용하면 두번째 파일의 첫행이 첫번째 파일의 첫행 뒤에 덧붙여 지는 방법으로 파일들이 혼합된다.

cut 지령을 사용하여 허락마당 혹은 1번 기호부터 10번 기호까지만을 추려 내고 그 결과를 ll.out에 보관시키자.

$ **ls -al | cut -c 1-10**

```
total 798
drwxrwxrwx
drwxrwxrwx
```

```
-rwxrwxrwx
-rwxrwxrwx
-rwxrwxrwx
drwxr-xr-x
-rwxrwxrwx
-rw-r--r--
-rwxrwxrwx
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-r--r--
-rw-rw-rw-
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rw-r--r--
-rwxrwxrwx
-rwxr-xr-x
-rw-r--r--
-rwxrwxrwx
drwxr-xr-x
-rwxrwxrwx
-rw-r--r--
-rw-r--r--
```

```
$ ls -al | cut -c 1-10 > ll.out
$
```

다음은 지령 paste를 사용하여 ll.out 파일에 보관되여 있는 허락들을 llsum.out 파일에 결합시켜 보자.

$ **paste llsum. out ll. out**

```
FILENAME              SIZE          total 792
README                 810          -rwxrwxrwx
backup_files          3408          -rwxrwxrwx
biography              427          -rwxrwxrwx
cshtest               1024          drwxr-xr-x
gkill                 1855          -rwxrwxrwx
gkill.out              191          -rw-r--r--
hostck                 924          -rwxrwxrwx
ifBtat                1422          -rwxrwxrwx
ifstat.int            2147          -rwxr-xr-x
ifstat.out             723          -rw-r--r--
introdos             54018          -rw-r--r--
introux              52476          -rwxrwxrwx
letter               23552          -rw-r--r--
letter.auto          69632          -rw-r--r--
letter.auto.recover  71680          -rw-r--r--
letter.backup        23552          -rw-r--r--
letter.lck              57          -rw-rw-rw-
letter.recover       69632          -rw-r--r--
ll.out                1057          -rw-r--r--
llsum                 1267          -rwxrwxrwx
llsum.orig            1267          -rwxr-xr-x
llsum.out             1657          -rw-r--r--
llsum.tomd.out        1356          -rw-r--r--
psg                    670          -rwxrwxrwx
psg.int                802          -rwxr-xr-x
psg.out                122          -rw-r--r--
sam_adduser           1010          -rwxrwxrwx
tdolan                1024          drwxr-xr-x
trash                 4554          -rwxrwxrwx
trash.out              329          -rw-r--r--
typescript             679          -rw-r--r--

$
```

이 실례는 llsum. out 의 파일이름과 파일크기마당 그리고 ll. out 의 허락마당을 결합하여 현시하고 있다.

만일 두개의 파일이 동일한 첫 마당을 가지고 있다면 join 지령으로 두개의 파일을 결합할수 있다.

아래에서는 paste 및 join 지령에 대한 개요를 보여 주고 있다.

paste - 파일들의 행들을 혼합한다.

선택항목들

-d list	list 는 렬들사이의 분리기호로서 행바꾸기기호 \n, 태브문자 \t 와 같은 특수한 탈퇴렬이 될수 있다.

join - 공통적인 열쇠마당을 가지는 두개의 미리 정돈된 파일들을 결합한다.

선택항목들

-a n	표준적인 출력을 생성하며 1 혹은 2 에서 결합될수 없는 매개 행에 대한 하나의 행을 생성한다.
-e string	빈 마당들을 string 으로 교체하여 출력한다.
-t char	char 를 마당분리기호로서 리용한다.

tr

tr 는 문자들을 변환하는 지령이다. 실례로 모든 소문자들을 대문자로 변경하려는 경우에 tr 지령을 사용할수 있다. 다음의 실례에서는 뒤붙이 "zip"를 가지는 파일들을 현시하며 그것을 대문자로 변경시키고 있다.

$ **ls -al *. zip**
fi1el. zip
fi1e2. zip
fi1e3. zip
fi1e4. zip
fi1e5. zip
fi1e6. zip
fi1e7. zip
$ **ls -al *. zip tr "[:lower:]m "[:upper:]"**
FILE1. ZIP
FILE2. ZIP
FILE3. ZIP
FILE4. ZIP
FILE5. ZIP

FILE6. ZIP
FILE7. ZIP
$

우의 실례에서처럼 문자들의 모임을 지적하는 경우에 괄호를 사용한다. 아래에서는 tr지령에 대한 개요를 주고 있다.

tr - 문자들을 변경시킨다.

선택항목들

-A	바이트별로 변경한다.
-d	규정된 문자들의 모든 발생을 제거한다.
[:class:]	소문자모임을 대문자모임으로 변경하는것과 같이 한 문자족을 다른 문자족으로 변경한다.

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 좀 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

cmp

cmp – 파일들을 비교한다.

---

cmp(1)

이름

cmp – 두 파일을 비교한다.

형식

cmp [-l] [-s] file1 file2

해설

cmp 는 두개의 파일들을 비교한다. 표준적으로 cmp 는 파일들이 동일하면 아무런 설명도 생성하지 않는다. 그러나 두 파일이 차이나면 차이가 발생하는 바이트와 행을 출력한다. 만일 하나의 파일이 다른것의 부분이라면 그 정보를 알려

준다.

선택항목들

-l　　매 차이점에 대한 바이트번호(10진수)와 차이나는 바이트내용(8진수)을 현시한다(바이트번호화는 1부터 시작한다.).

-s　　차이나는 파일에 대한 귀환코드만을 현시한다.

외부적영향

환경변수들

LANG은 현시되는 통보문의 언어를 결정한다. 만일 LANG이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 cat는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

진단

cmp는 다음의 탈퇴값들을 귀환한다.

0　　파일들은 동일하다.
1　　파일들은 동일하지 않다.
2　　접근불가능 혹은 인수가 모자란다.

cmp는 file1(file2)의 끝까지 비교가 성과적으로 진행되면 다음의 경고를 현시한다.

cmp: EOF on file1(file2)

관련항목

comm(1), diff(1)

표준일치

cmp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

comm

comm – 정돈된 파일에 대한 세개의 부분으로 된 출력을 내보낸다.

---

comm(1)

이름

comm – 두개의 정돈된 파일에 대한 공통적인 행들을 선택 혹은 거절한다.

형식

comm [-[123]] file1 file2

해설

comm 은 file1 과 file2 를 읽어서 세개의 부분으로 된 출력을 생성한다. 이때 두개의 파일은 이미 증가방향으로 정돈되여 있어야 한다.

첫번째 부분: file1 에만 나타나는 행들
두번째 부분: file2 에만 나타나는 행들
세번째 부분: 두개의 파일에 다 나타나는 행들

만일 -가 file1 이나 file2 에서 리용된다면 표준입력장치가 리용된다.

선택항목 1, 2, 3 은 대응하는 부분에 대한 출력을 금지한다. 즉 comm -12 는 두개의 파일에 공통인 행들만 출력한다. comm -23 은 첫번째 파일에만 존재하는 행들만 현시하며 comm -123 은 아무것도 현시하지 않는다.

외부적영향

환경변수들

LC_COLLATE 는 입력파일로부터 comm 이 기다리는 렬을 결정한다.

LC_MESSAGES 는 통보가 현시되는 언어를 결정한다. 만일 이 변수가 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 리용된다. LC_COLLATE 가 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 comm 은 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

다음의 실례에서는 file1 과 file2 가 LC_COLLATE 나 LANG 환경변수에 의하여 정의된 코드모임으로 정돈되여 있다고 가정한다.

file1 과 file2 에 공통적인 행들을 모두 출력하자면 다음과 같이 하여야 한다.

comm -12 file1 file2

file1 에만 나타나는 모든 행들을 출력하자면 다음과 같이 하여야 한다.

comm -23 file1 file2

file2 에만 나타나는 모든 행들을 출력하자면 다음과 같이 하여야 한다.

comm -13 file1 file2

관련항목

cmp(1), diff(1), sdiff(1), sort(1), uniq(1)

표준일치

comm: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

cut

cut – 파일안의 행들로부터 선택된 마당들을 자른다.

---

cut(1)

이름

cut – 파일의 매개 행에서 선택된 마당들을 자른다.(추려 낸다.)

형식

cut -c list [file …]
cut -b list [-n] [file]
cut -f list [-d char] [-s] [file …]

해설

cut 는 파일의 매개 행으로부터 마당들을 추출하거나 표로부터 렬들을 추려 낸다. list 에서 규정한 마당들은 고정길이를 가지거나 그 길이는 행에 따라 변할수 있다. 또한 마당들은 태브기호와 같은 마당분리기호로 구분될수 있다. cut 는 파일이 주어 지지 않고 표준입력장치가 리용되는 경우 려과기로서 사용할수 있다.

단일바이트기호모임을 처리할 때 -c 및 -b 들은 동등하며 동일한 결과를 생성한

다. 다중바이트기호모임을 처리할 때 -b 및 -c 들은 함께 리용되며 그것들이 결합된 동작은 대단히 류사하지만 -c 와는 같지 않다.

선택항목들

list 반점으로 구분된 옹근수바이트(-b), 문자(-c) 혹은 마당(-f)번호들이 증가순서로 되여 있는 목록이다. 실례로

1, 4, 7	1, 4, 7번 위치들
1-3, 8	1번부터 3번까지 그리고 8번 위치
-5, 10	1번부터 5번까지 그리고 10번 위치
3-	3번부터 마지막위치까지

-b list 바이트목록의 자르기이다.

-c list 기호목록의 자르기이다.(-c 1-72 는 매개 행에서 첫 72 개의 기호들을 추려 낸다.)

-f list 마당목록의 자르기이다. 실례로 -f 1, 7 은 첫번째와 일곱번째 마당만을 추려 낸다. 마당분리기호가 없는 행들은 -s 가 규정되여 있지 않으면 그대로 통과된다(표의 머리부에 대하여 효과적이다.).

-d char char는 마당분리기호로서 리용한다(-f때에만). 기정값은 태브이다. 공백이나 쉘에서 특수한 의미를 가지는 기호들은 " " 안에 넣어야 한다.

-n 기호들을 자르지 않는다. 목록안에 있는 어떤 령역의 웃끝이 기호의 마지막바이트가 아니라면 그 기호는 출력에 관계되지 않는다. 그러나 령역의 아래끝이 기호의 첫 바이트가 아니라면 그 기호는 출력에 포함된다.

-s -f를 리용했을 때 분리기호가 없는 행들은 금지된다. -s가 규정되지 않았으면 분리기호가 없는 행들은 출력에서 변경없이 나타난다.

외부적영향

암시

정규식으로 규정된 본문을 파일로부터 추려 내려면 grep 를 리용한다. 파일들을 행단위로 혼합하여 렬형식으로 출력하려면 paste 를 리용한다. 표에서 렬들을 줄맞추기하려면 cut 와 paste 를 리용한다(grep(1)과 paste(1)을 참고).

환경변수들

LC_CTYPE 는 본문에 대한 단일/다중바이트기호들의 해석을 결정한다.
만일 LC_CTYPE 가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이

기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 cut 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

-d 로 규정된 분리기호는 단일바이트기호여야 한다. 그밖의 경우에는 단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

사용자 ID 에 대응하는 파일을 사용자이름으로 암호화하자면 다음과 같이 하여야 한다.

```
cut -d : -f 1, 5 /etc/passwd
```

환경변수이름을 현재의 가입이름으로 설정하자면 다음과 같이 하여야 한다.

```
name=`who am i | cut -f 1 -d " "`
```

임의의 길이의 행들을 포함하는 source 파일을 두개의 파일로 변환하자면 다음과 같이 하여야 한다(file1 은 첫 500byte 를 포함하고 file2 는 매행의 나머지부분을 포함하도록 한다. 500byte 기호는 다중바이트기호로 한다.).

```
cut -b 1-500 -n source > file1
cut -b 500 -n source > file2
```

진단

line too long

행길이는 행바꾸기기호까지 포함하여 LINE_MAX 만큼의 기호 혹은 마당들을 넘지 말아야 한다.

bad list for b/c/f option

-b, -c, -f 가 없거나 부정확하게 규정된 목록을 지정한다. 만일 찾으려는 목록보다 마당개수가 작다면 오유는 일어 나지 않는다.

no fiels

목록이 비였다.

경고

cut 는 태브를 전개하지 않기때문에 태브가 필요한 경우에는 본문을 expand(1)로 파이프련결하시오.

<Backspace>기호들은 다른 기호로 처리된다. cut 처리에 앞서 <Backspace>기호들을 소거하려면 fold(1)과 col(1)을 사용하시오.

저자

cut는 OSF와 HP에서 개발하였다.

관련항목

grep(1), paste(1)

표준일치

cut: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

diff

diff - 파일 및 등록부를 비교한다.

---

diff(1)

이름

diff - 서로 다른 파일 및 등록부의 비교자이다.

형식

diff [-C n] [-S name] [-lrs] [-bcefhintw] dir1 dir2
diff [-C n] [-S name] [-bcefhintw] dir1 dir2
diff [-d string] [-biw] file1 file2

해설

등록부비교

두개의 인수가 등록부일 때 diff는 등록부의 내용을 이름별로 정돈하고 매개 등록부에 대하여 생성되는 본문파일(이름은 같지만 서로 다른 파일)에 대한 diff알고리듬을 수행한다(아래에서 설명). 차이나는 2진파일들, 공통적인 보조등록부들, 하나의 등록부에만 나타나는 파일들이 렬거된다. 등록부들을 비교할 때 다음과 같은 선택항목들이 인식된다.

-l 긴 출력형식을 규정한다. 차이나는 매개 본문파일은 pr로 파이프련결되여 페지단위로 현시된다. 다른 차이들은 본문파일 차이들이 모두 출력된 뒤에 개괄된다.

-r 공통적인 보조등록부들에 diff를 재귀적으로 적용한다.

-s 동일한 파일들만을 현시한다.

-S name

주어 진 파일이름으로부터 시작하여 정돈된 등록부의 중간에서부터 diff를 수행시킨다.

파일비교

등록부비교로 차이나는 본문파일들을 비교할 때 diff는 어느 행을 변경시켜야 그것들이 동등해 지겠는가를 알려 준다. diff는 보통 파일차이들을 포함하는 가장 작은 모임을 찾아 낸다. file1이나 file2가 등록부가 아니라면 그것들은 -s로 규정될수 있다. 즉 표준입력장치가 리용될수 있다. file1이 등록부인 경우에 그 등록부안에 있는 어떤 파일이 file2로 리용될수 있다.

출력형식에 대해서는 여러가지 선택항목들이 존재한다. 표준출력형식은 다음과 류사한 행들을 포함한다.

n1 a n3, n4
n1, n2 d n3
n1, n2 c n3, n4

이 행들은 file1을 file2로 변환하는 ed지령과 비슷하다. 글자뒤에 있는 수자들은 file2에 속하게 된다. a를 d로 바꾸는것으로써 file2을 file1로 바꾸게 할수 있다. ed에서처럼 n1=n2 혹은 n3=n4인 동등한 쌍들은 하나의 수로 생략된다. 이러한 행들뒤에는 기호 "<"로 표식된 첫번째 파일에 있는 모든 행들이 오며 또 그뒤에는 기호 ">"로 표시된 두번째 파일에 있는 모든 행들이 온다.

선택항목 -b, -w, -i, -t 들을 제외한 다음의 선택항목들은 호상 배타적으로 리용된다.

-e file1로부터 file2를 다시 창조하는데 ed편집기의 적중한 a, c, d 지령들을 생성한다. 나머지 지령들은 -e로서 등록부를 비교할 때 출력에 추가된다. 그리하여 결과는 두개의 등록부에서 공통적인 본문파일들을 dir1의 상태로부터 dir2의 상태로 변환하기 위한 쉘형식으로 된다(sh-bourne(1)을 참고).

-f -f는 -e 선택항목과 비슷한 형식을 출력한다. ed의 견지에서는 효과적이지 못하지만 사용자의 견지에서는 보다 읽기가 쉽다.

-n -e와 비슷한 형식이지만 반대순서이다. 그리고 삽입 혹은 제거지령으로 변경된 행들을 계산한다. 이것은 rcsdiff에서 리용되는 형식이다(rcsdiff(1)을 참고).

-c 세개의 행으로 된 문맥으로 차이점들을 현시한다. -c는 출력형식을 일부 변경시킨다. 즉 먼저 포함된 파일들을 식별하고 그뒤에 창조날자, 12개의 "*"를 포함하는 변화된 행들을 내보낸다. file1로부터 제거된 행들에는 -, file2에 추가된 행들

에는 +부호를 붙인다. 한 파일로부터 다른 파일로 변경된 행들에는 "!"를 붙인다. 파일안에서 변경된 내용들은 세개의 행단위로 그룹화되여 출력된다.

-C n　　문맥이 n개 행이라는것을 제외하고는 -c와 류사하다.

-h　　매우 빠른 비교를 진행한다. 변경내용이 작고 잘 구분되여 있을 때 사용한다.

-D string

file1 과 file2 를 혼합하여 표준출력으로 내보내는 방안이다. string의 정의를 하지 않고 결과를 콤파일하는것은 file1 을 콤파일하는것과 동등하며 정의된 string 으로 결과를 콤파일하는 것은 file2 를 콤파일하는것과 동등하도록 C 의 **전처리기** (Preprocessor)를 포함한다.

-b　　마지막공백들을 무시하며 공백이 있는 다른 기호렬들은 그대로 처리한다.

-w　　모든 공백들을 무시한다.

-i　　대소문자차이를 무시한다.

-t　　출력행에서 태브를 현시한다. 표준적인 방식과 -c 방식의 출력은 한개이상의 문자들을 매행의 앞에 추가한다. 그러나 이 선택항목은 초기원천파일 그대로 출력하게 한다.

외부적영향

환경변수들

LANG 은 LC_ALL 과 LC_으로 시작되는 대응하는 환경변수가 정의되지 않았을 때 지역분류를 위한 그 지역을 결정한다. 만일 LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

LC_CTYPE 는 diff 지령에 대한 공백기호, 본문의 단일/다중바이트기호들에 대한 해석을 결정한다.

LC_MESSAGES는 현시되는 통보문의 언어를 결정한다.

어떤 국제화변수가 틀린 설정을 포함하면 diff 및 diffh 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

diff 및 diffh 가 다중바이트 <Alt>기호를 인식하지 못한다는것을 제외하고는 단일바이트 및 다중바이트기호코드모임이 보장된다.

귀환값

diff는 다음과 같은 탈퇴값들을 가지고 귀환한다.

0	아무런 차이도 발견하지 못했다.
1	차이가 발견되였다.
>1	오유가 발생하였다.

실례

다음의 지령은 script라는 하나의 스크립트파일을 창조한다.

```
diff -e x1 x2 > script
```

w는 파일을 보관하기 위하여 script의 끝에 추가된다.

```
echo w >> script
```

script파일은 파일 x1로부터 ed편집기를 리용하여 파일 x2을 창조하는데 리용될 수 있다.

```
ed x1 < script
```

다음의 지령은 두개의 행단위로 차이나는 출력통보를 생성한다.

```
diff C2 x1 x2
```

다음의 지령은 모든 공백과 태브들 그리고 대소문자의 차이를 무시한다.

```
diff -wi x1 x2
```

경고

-e 혹은 -f 로서 생성된 스크립트들을 편집하는것은 단순점(.)으로 구성된 행들을 창조하는것과 마찬가지로 간단하다.

-b, -w, -i 들로 등록부들을 비교할 때 diff는 먼저 cmp와 동일한 방식으로 파일들을 비교하고 그다음에 그것들이 동등하지 않으면 diff 알고리듬을 수행시킨다. 이렇게 하면 공백렬이나 대소문자차이와 같은것들을 제외하고는 서로 동등한 파일들을 비교할 때 출력결과를 단순하게 현시할수 있다.

표준적인 알고리듬은 파일크기의 6배나 되는 기억공간을 요구한다. 충분한 기억이 보장되지 못하면 -h 혹은 bdiff는 리용될수 없다(bdiff(1)을 참고).

-r 로 등록부를 수행시킬 때 diff는 재귀적으로 부분나무들을 감소시킨다. 깊은 다중준위의 등록부들을 비교할 때에는 보다 많은 기억공간이 필요하다.

저자

diff는 AT&T, 캘리포니아종합대학, 버클리, HP에 의하여 개발되였다.

파일

/usr/lbin/diffh 선택항목 -h 에 의하여 리용된다.

관련항목

bdiff(1), cmp(1), comm(1), diff3(1), dircmp(1), ed(1), more(1), nroff(1), rcsdiff(1), sccsdiff(1), sdiff(1), terminfo(4)

표준일치

diff: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

dircmp

dircmp - 등록부들을 비교하고 결과를 출력한다.

---

dircmp(1)

이름

dircmp - 등록부비교.

형식

dircmp [-d] [-s] [-wn] dir1 dir2

해설

dircmp 는 dir1 과 dir2 를 비교하고 그 등록부들의 내용에 대한 정보를 태브를 리용하여 생성한다. 모든 선택항목들에 대하여서는 매개 등록부에 고유한 정돈된 목록이 출력된다. 선택항목이 하나도 없으면 동일한 내용을 가지는 두개의 등록부에서 공통인 파일이름들을 정돈하여 출력한다.

선택항목들

-d 두개의 등록부에서 동일한 이름을 가진 파일들의 내용을 비교하고 두개의 파일이 같아 지자면 무엇이 변경되여야 하는가를 알려 준다.

-s 동일한 파일들에 대한 통보를 금지한다.

-wn 출력행의 너비를 n 개의 기호로 규정한다. 기정값은 72 이다.

외부적영향

환경변수들

LC_COLLATE 는 출력이 정돈되는 순서를 결정한다.

만일 LC_COLLATE 가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG

의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 dircmp 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

두개의 등록부를 비교하여 그 등록부들이 서로 같게 만들자면 다음과 같이 하여야 한다.

```
dircmp -d slate sleet
```

관련항목

cmp(1), diff(1)

표준일치

dircmp: SVID2, SVID3, XPG2, XPG3

join

join – 파일들에서 행단위로 두개의 관계를 결합시킨다.

---

join(1)

이름

join – 관계형자료기지연산자.

형식

join [options] file1 file2

해설

join 은 file1 과 file2 의 행에서 규정된 두개의 관계들을 결합시킨다. file1 혹은 file2 가 -라면 표준입출력장치가 리용된다.

file1 과 file2 는 결합되여야 할 마당에 관하여 증가순서로 미리 정돈되여 있어야 한다. 보통 이 마당은 매개 행에서 첫 마당이다.

출력은 결합마당이 동등한 file1 과 file2 의 행들의 결합쌍을 하나의 행으로 내보낸다. 보통 출력행은 공통마당, file1 의 나머지마당들, file2 의 나머지마당들로

구성된다.

표준적인 입력마당분리기호들은 공백, 태브, 행바꾸기기호들이다. 이 경우에 다중마당분리기호들이 하나의 마당분리기호로 되면 뒤에 있는 분리기호들은 무시된다. 기정출력마당분리기호는 공백이다.

아래의 선택항목들에서 리용하는 n 인수는 file1 혹은 file2 를 가리키는 1 혹은 2 이다.

선택항목들

-a n 표준적인 출력외에 filen 에 있는 결합불가능한 행을 추가적으로 생성한다.

-e s 빈 출력마당들을 기호렬 s 로 교체한다.

-j m 마당 m 에 대한 결합을 진행한다. 이 선택항목과 아래의 두개 선택항목들은 이전방안과의 호환성을 위하여 제공된다.

-j1 m file1 의 마당 m 에 대한 결합을 진행한다.

-j2 m file2 의 마당 m 에 대한 결합을 진행한다.

-o list 매개 출력행은 list 에서 규정한 마당들을 포함한다. 행의 매개 원소들은 n.m 형식을 가진다. 여기서 n 은 파일번호이고 m 은 마당번호이다. 공통마당은 출력되지 않는다.

-t c 문자 c 를 입력 및 출력의 분리기호로 리용한다.

-v n 표준적인 출력대신에 규정된 파일안에 있는 매개 결합불가능한 행들만을 출력한다.

-v file_number
표준출력대신에 file_number 1 혹은 2 에서 짝 지어 지지 않은 행들만 생성한다.

-1 f file1 의 마당 f 에 대한 결합을 진행한다. 마당들은 1 부터 번호가 붙는다.

-2 f file2 의 마당 f 에 대한 결합을 진행한다. 마당들은 1 부터 번호가 붙는다.

외부적영향

환경변수들

LC_COLLATE 는 입력파일로부터 join 이 기다리는 렬을 결정한다.

LC_CTYPE 는 입력마당분리기호로 되는 <Alt>기호를 결정하며 파일안의 자료에 대한 단일다중바이트기호들의 해석을 결정한다. 또한 -t 를 통하여 정의된 분리

기호가 단일바이트기호인가 혹은 다중바이트기호인가를 결정한다.

만일 LC_COLLATE 나 LC_CTYPE 가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 join 은 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

다중바이트기호의 파일이름이 보장되지 않는다는것을 제외하고는 단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

다음의 지령은 암호파일과 그룹파일을 수값그룹 ID 에 기초하여 결합하며 가입이름, 그룹이름, 가입등록부를 출력한다. 여기서 파일들은 환경변수 LC_COLLATE 혹은 LANG 에 의하여 정의된 수직렬로 그룹 ID 마당에 기초하여 정돈되여 있다고 가정한다.

join -l 4 -2 3 -o 1.1, 2.1, 1.6 -t : /etc/passwd /etc/group

다음의 지령은 두개의 정돈된 파일 sf1 와 sf2 에서 동일한 첫 마당을 가지는 행들의 가능한 결합모두를 현시한다. 여기서 매개 행은 sorted_file1 로부터 첫번째와 세번째 마당이 구성되며 sorted_file2 로부터 두번째와 네번째 마당이 구성된다.

join -j1 l -j2 1 -o 1.1, 2.2, 1.3, 2.4 sorted_file1 sorted_file2

경고

표준마당분리를 리용한 코드모임은 -b 로 정돈된것이며 -t 로 정돈된것은 평범한 정돈이다.

join, sort, comm, uniq, awk 를 사용하는 습관은 서로 각이하다.

수값으로 된 파일이름들은 -o 가 리용될 때 파일이름을 렬거하기전에 즉시로 충돌을 일으킨다.

저자

join 은 OSF 와 HP 에서 개발하였다.

관련항목

awk(1), comm(1), sort(1), uniq(1)

표준일치

join: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

paste

paste － 파일의 행들을 혼합한다.

paste(1)

이름

paste － 여러 파일로부터 동일한 행들을 혼합하거나 한 파일의 행들을 순차적으로 렬거한다.

형식

paste file1 file2 …
paste -d list file1 file2 …
paste -s [-d list] file1 file2 …

해설

첫 두개의 형식에서 paste 는 주어 진 입력파일 file1, file2 … 로부터 대응하는 행들을 련결한다. 이때 매개 파일은 표에서의 렬처럼 처리되며 수평으로 련결된다(병렬혼합). -s 가 있으면 입력파일의 행들을 순차적으로 혼합한다(직렬혼합). 두가지 경우에 모두 태브 혹은 list 에서 규정한 문자들로 결합된다. 출력은 표준출력장치로 내보낸다. paste 는 파이프의 앞단계에서 리용될수 있으며 또한 려과기로 리용될수도 있다.

선택항목들

-d 이 선택항목이 없으면 마지막파일(-s 가 있는 경우에는 마지막행)을 제외한 파일들에서 모든 행바꾸기기호들은 태브기호로 바뀌운다. 그러나 이 선택항목은 태브기호를 한개이상의 <Alt>기호로 바꿀수 있게 한다.

list -d 뒤에 오는 한개이상의 기호들로 이루어 진 목록으로서 표준적인 행결합기호인 태브를 이 기호들로 바꾼다. 이 목록은 특수한 탈퇴렬들인 \n, \\, \0 과 같은것들을 포함할수 있다.

-s 매개 입력파일들로부터 행들을 순차적으로 혼합한다. -d 가 규정되지 않았으면 태브를 표준적인 결합기호로 사용한다. 목록에 무관계하게 파일의 제일 마지막기호는 행바꾸기기호여야 한다.

\- 표준입력장치로부터 행을 읽어 어떤 파일에 결합시킬수 있게 한다.

외부적영향

환경변수들

LC_CTYPE 는 본문의 단일/다중바이트기호들의 해석을 위한 지역을 결정한다.

LC_MESSAGES 는 현시되는 통보문의 언어를 결정한다.

만일 LC_CTYPE 혹은 LC_MESSAGES 가 설정되지 않았거나 빈 기호렬로 되여 있으면 LANG 이 기정값으로 리용된다. 또한 LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 paste 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

귀환값

0	성과적으로 완성
>0	오유발생

실례

등록부를 한개 렬로 현시하자면 다음과 같이 하여야 한다.

```
ls | paste -d " " -
```

등록부를 네개 렬로 현시하자면 다음과 같이 하여야 한다.

```
ls | paste  - - - -
```

행들의 쌍을 행으로 결합하여 현시하자면 다음과 같이 하여야 한다.

```
paste -s -d "\t\n" file
```

주의

pr -t - …는 paste 와 류사하게 작업하지만 페지륜곽선을 주기 위하여 외부적인 공백, 태브, 행바꾸기들을 창조한다.

진단

too many files

-s 를 제외하고 OPEN_MAX -3 에서 규정한것보다 입력파일이 많지 말아야 한다.

저자

paste 는 OSF 와 HP 에 의하여 개발되였다.

관련항목

cut(1), grep(1), pr(1)

표준일치

paste: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

sort

sort – 파일들의 내용을 정돈한다.

---

sort(1)

이름

sort – 파일들을 정돈하거나 혼합한다.

형식

sort [-m] [-o output] [-bdfinruM] [-t char] [-k keydef] [-y [kmem]] [-z recsz] [-T dir] [file …]

sort [-c] [-AbdfinruM] [-t char] [-k keydef] [-y [kmem]] [-z recsz] [-T dir] [file …]

해설

sort는 다음과 같은 기능들중의 한가지를 수행한다.

- 주어 진 파일들의 모든 행들을 함께 정돈하고 그 결과를 규정된 출력장치에 현시한다.
- 이미 정돈된 주어 진 파일들의 모든 행들을 혼합하여 규정된 출력장치에 현시한다.
- 하나의 입력파일이 정확히 정돈되여 있는가를 검사한다.

-가 파일이름으로 사용되고 다른 입력파일들이 하나도 규정되지 않았다면 표준입력장치로부터 읽는다.

비교는 입력의 매개 행에서 추려 낸 한개이상의 정돈열쇠에 의하여 진행된다. 순서화는 현재지역 지대의 코드모임을 리용하여 사전의 자모순서로 진행한다. 지역이 규정되지 않거나 POSIX 지역으로 설정되여 있으면 순서화는 기계코드모임의 바이트순서로 진행한다. 만일 지역이 다중바이트기호를 포함한다면 다중바이트기호들앞에 단일바이트기호들이 기계코드에 의하여 대응되게 된다.

동작선택항목들

-A 바이트별로 정돈한다. 일부 체계들에서 확장기호들은 부의 값으로 고찰하여 ASCII 기호들보다 앞서도록 정돈한다. 만일 ASCII 기호들을 비 C/POSIX 지역에서 정돈한다면 이 기발은 보다 빠르게 수행된다.

-c 정돈해야 할 하나의 입력파일을 검사한다. 출력은 생성되지 않으며 결과를 반영하는 귀환코드가 설정된다.

-m 혼합만을 한다. 즉 입력파일들은 이미 정돈되였다고 가정한다.

-o output
출력파일을 규정한다. 즉 표준출력장치가 아닌것을 지적할수 있다. 이 파일은 입력파일중의 하나로도 될수 있다.

-u 동일한 열쇠를 가지는 행 하나만을 제외하고 나머지는 모두 금지한다. -c 와 함께 리용되면 정돈상태의 검사와 함께 중복열쇠가 없는가를 더 검사한다.

-y [kmem]
kmem 은 sort 가 수행될 때 리용하는 주기억의 키로바이트수(KB)이다. 이 선택항목이 지적되지 않으면 sort 는 체계의 표준적인 기억량을 가지고 수행된다. kmem 을 생략하고 -y 만 주면 최대기억량을 가지고 정돈을 진행한다.

-z recsz 가장 긴 행의 크기를 기록하여 완충기를 규정하도록 한다. 이 선택항목을 규정하지 않으면 체계의 표준적인 크기로 완충기가 규정되는데 완충기크기보다 더 긴 행들은 잘리워 정돈된다.

-T dir sort 가 리용하는 림시등록부 dir 를 지적한다. 기정값으로 주어 지는 림시등록부는 환경변수 TMPDIR 에 의하여 규정될수도 있고 /var/tmp, /tmp 등이 될수도 있다.

순서화규칙선택항목들

-d 거의 사전식순서로서 자모문자들과 LC_CTYPE 에서 정의된 공백기호들만이 의미를 가지게 한다.

XPG4 에서만은 -i 혹은 -n 이 적용되는 정돈열쇠들에 대한 작용이 정의되지 않는다.

-f 문자들을 겹치게 한다. 즉 비교가 진행되기전에 모든 소문자들을 LC_CTYPE 에서 정의된 대응하는 대문자들로 변환한다.

-i 비수값적인 비교에서 LC_CTYPE 에 의하여 정의된 모든 인쇄불가능한 기호들을 무시한다. 즉 ASCII 기호모임에서 8 진수로 001 부터 037 까지 및 0177 까지를 무시한다.

-n　　정돈열쇠를 선택적인 공백, 선택적인 -부호, 령이나 선택적인 밀수기호들, 선택적인 여러개의 분리기호들만으로 제한한다. 밀수와 분리기호들은 LC_NUMERIC 에서 정의된다. 마당은 수값으로 정돈된다. 빈 수값마당은 산수적인 령으로 고찰된다. 선택항목 -n 은 선택항목 -b 를 암시적으로 적용하고 있다.

-r　　비교순서를 거꿀순서로 한다.

-M　　월별비교를 한다. 마당에서 몇개의 비지 않은 첫 부분의 문자들을 대문자로 변환하고 langinfo(5)의 항목들인 ABMON_1 < ABMON_2 < … < ABMON_12 와 비교한다. 틀린 마당은 ABMON_1 기호렬보다 작다고 본다. 실례로 JAN < FEB < … < DEC 로 비교된다. 선택항목 -M 은 선택항목 -b 를 암시적으로 적용하고 있다.

마당분리선택항목들

-t char　　char 는 마당분리기호이다. -t 가 규정되지 않으면 공백기호들이 표준적으로 리용된다.

-b　　정돈열쇠의 앞과 뒤에 놓이는 공백기호들을 무시한다.

정돈선택항목

-k keydef

keydef 는 정돈열쇠를 정의하는 인수로서 형식은 다음과 같다.

field_start [type] [, field_end[type]]

이것은 열쇠마당이 field_start 로 시작하여 field_end 로 끝난다는것을 의미한다. field_start 와 field_end 에 놓이는 기호들은 열쇠마당에 포함된다. field_end 가 없으면 행끝을 의미한다. 마당들과 기호들은 1 부터 순서화된다. 그러나 절대적정돈열쇠에서는 0 부터 시작한다.

field_start 와 field_end 규정에서는 마당의 표시, 마당분리기호나 행바꾸기기호앞에 놓이는 최소기호렬을 포함한다. 표준적으로 첫 공백렬의 공백이 마당분리기호로 작용한다. 공백렬의 모든 공백들은 다음마당의 부분으로 고찰된다.

field_start 와 field_end 는 m.n 형식을 가진다. 그리고 그 뒤에 선택적으로 b, d, f, i, n, r, M 들이 한개이상 놓일수 있다. 이 선택항목들의 작용은 그 열쇠에만 작용한다.

m.n 으로 규정된 field_start 지정은 m 번째 마당의 n 번째 문자를 의미한다. n 을 생략하면 m.1 을 의미한다. 그 경우에 -b 가 있으면 n 은 m 번째 마당에서 첫 비공백기호(처음으로 나타나는 공백이 아닌 기호)로 된다.
m.n 으로 규정된 field_end 지정은 m 번째 마당의 n 번째 기호를 의미한다. n 이 없으면 m 번째 마당의 마지막기호를 의미한다. 그 경우에 -b 가 있으면 n 은 m 번째 마당에서 첫 비공백기호로 된다.

여러개의 -k 가 가능하며 그것들이 놓이는 순서가 중요하다. 최대로 9 개의 -k 가 놓일수 있다. -k 가 규정되지 않으면 표준적인 정돈열쇠는 전체 행으로 된다. 여러개의 정돈열쇠들이 있을 때 뒤에 있는 열쇠들이 앞선 열쇠들보다 후에 비교된다.

-k 는 [+pos1 [+pos2]] 표시법으로 절대적인 field_start 와 field_end 를 규정할수 있다. 완전히 규정된 +w.x - y.z 는 다음의 것과 동등하다.

-k w+1.x+1, y.0 （ z = = 0 인 때）
-k w+1.x+1, y+1.z （ z > 0 인 때）

절대적인 정돈열쇠

+pos1 -pos2 형식은 정돈열쇠가 pos1 에서 시작하여 pos2 에서 끝난다는것을 의미한다. pos1 과 pos2 에 있는 문자들은 정돈열쇠에 포함된다. -pos2 이 없으면 행끝을 의미한다.

pos1 및 pos2 의 규정에서는 마당의 표시, 마당분리기호 혹은 행바꾸기기호앞에 놓이는 최소기호렬을 포함할수 있다. 표준적으로는 공백렬의 첫 공백이 마당분리기호로 작용한다. 공백렬의 모든 공백들은 다음마당의 부분으로 고찰된다.

pos1 과 pos2 는 m.n 형식을 가진다. 그리고 그뒤에 선택적으로 b, d, f, i, n, r, M 들이 한개이상 놓일수 있다. +m.n 으로 규정되는 시작위치는 m+1 번째 마당의 n+1 번째 기호를 의미한다. n 을 생략하면 m.0 을 의미한다. 그 경우에 -b 가 있으면 n 은 m+1 번째 마당의 첫 비공백기호로 된다. 즉 +m.0b 는 m+1 번째 마당의 첫번째 비공백기호를 참조한다.

-m.n 형식의 마지막지정은 m 번째 마당의 마지막기호뒤에 있는 n 번째 기호(분리기호까지 포함)를 의미한다. n 이 없으면 즉 .0 은 m 번째 마

당의 마지막기호를 의미한다. 그 경우에 -b 가 있으면 n 은 m+1 번째 마당에서 마지막공백으로 된다. 즉 -m.1b 는 m+1 번째 마당의 첫 비공백기호를 참조한다.

외부적영향

환경변수들

LC_COLLATE 는 정돈에 쓰이는 기정의 순서화규칙을 결정한다.

LC_CTYPE 는 본문자료의 바이트렬을 기호들(즉 인수들과 입력파일들에서 단일 및 다중바이트문자들)로서 해석하기 위한 지역과 -b, -d, -f, -i, -n 들에 대한 기호분류의 방법을 결정한다.

LC_NUMERIC 는 -n 에 대하여 밑수와 여러개의 분리기호들에 대한 정의를 결정한다.

LC_TIME 은 -M 에 대하여 월이름들을 결정한다.

LC_MESSAGES 는 현시되는 통보문의 언어를 결정한다.
LC_ALL 은 기타 국제화변수들의 모든 값을 무시하기 위한 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

LANG 은 설정되지 않았거나 비여 있는 국제화변수에 대한 기정값을 제공한다. LANG 이 결정되지 않았거나 null 이면 "C"가 기정값으로 리용된다(lang(5)을 참고). 어떤 국제화변수가 틀린 설정을 포함하면 sort 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

두번째 마당을 정돈열쇠로 하여 파일 infile 의 내용을 정돈하려면 다음과 같이 하여야 한다.

```
sort -k 2, 2 infile
```

두번째 마당의 첫 두개 문자를 정돈열쇠로 하여 file1 과 file2 의 내용을 거꿀순서로 정돈한 결과를 outfile 에 넣으려면 다음과 같이 하여야 한다.

```
sort -r -o outfile -k 2.1, 2.2 file1 file2
```

file1 과 file2 의 내용을 네번째 마당의 첫 비공백기호를 정돈열쇠로 하여 거꿀순서로 정돈하려면 다음과 같이 하여야 한다.

```
sort -r -k 4.1b, 4.1b file1 file2
```

수값으로 된 사용자 ID(세번째 마당)에 의하여 정돈된 암호파일(/etc/passwd)을 출력하려면 다음과 같이 하여야 한다.

```
sort -t : -k 3n, 3 /etc/passwd
```

이미 정돈되여 있는 파일 infile 에서 세번째 마당과 동일한 마당을 가지는 행의 첫 발생만을 출력하려면 다음과 같이 하여야 한다.

```
sort -mu -k 3, 3 infile
```

진단

0 모든 입력파일들이 성과적으로 출력되였을 때 혹은 -c 가 규정되고 입력파일이 정확히 정돈되여 있었을 때 이 값이 귀환된다.

1 이 값은 -c 에 대하여 파일들이 정돈되여 있지 않았을 때 혹은 -c 와 -u 가 규정되고 동등한 열쇠를 가지는 두개의 입력행들이 발견되였을 때 귀환된다. 이 귀환값은 -c 가 없으면 귀환하지 않는다.

>1 한개이상의 입력행들이 너무 길어서 오유가 발생하였을 때 귀환되는 값이다.

입력파일의 마지막행에 행바꾸기기호가 없을 때 sort 는 그것을 추가하고 경고통보를 내보낸 다음 계속한다.

규정된 언어에 대한 코드모임을 포함하는 표를 접근할 때 오유가 발생하면 sort 는 경고통보를 내보내고 POSIX 지령을 기정으로 설정한다.

-d, -f, -i 들이 다중바이트기호를 가지는 언어에서 규정되면 경고통보를 내보내고 그 선택항목을 무시한다.

경고

마당안에 있는 기호들과 마당의 순서화는 표준적으로 POSIX 로 한다. HP-UX 9.0 판부터는 1 부터 순서화하며 그이전 방안들은 0 부터 순서화한다.
-t 에 의하여 규정된 마당분리기호는 그것이 단일바이트기호일 때에만 인식된다.

다중바이트기호가 아닌 기호들의 형은 자모, 수자, 공백들로 분류된다. 다중바이트기호로 된 언어에서 모든 기호들은 서로 결합되여 있다.

파일

/var/tmp/stm???

/tmp/stm???

저자

sort는 OSF와 HP에 의하여 개발되였다.

관련항목

comm(1), join(1), uniq(1), collate8(4), envion(5), hpnls(5), lang(5)

표준일치

sort: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

tr

tr − 선택된 기호들을 치환한다.

---

tr(1)

이름

tr − 기호들을 변환한다.

형식

tr [-Acs] string1 string2

tr -s [-Ac] string1

tr -d [-Ac] string1

tr -ds [-Ac] string1 string2

해설

tr는 표준입력장치에서 주어 진 기호들을 교체하거나 제거한 다음 그 결과를 표준출력장치에로 보낸다. string1은 대응하는 string2로 교체된다.

선택항목들

-A　바이트별로 변환한다. 이 경우에 확장기호들은 보장되지 않는다.

-c　string1 안의 기호들의 모임을 보충한다. 이 모임은 LC_CTYPE 의 현재 설정에 의하여 정의되는것으로서 현재의 기호모임에 있는 모든 기호들의 모임이다. 이러한 기호들은 LC_COLLATE 의 현재 설정에 의하여 정의되는 코드모임안에 들어 있다.

-d　string1 에서 규정된 입력기호들 혹은 조사성분들의 모든 발생을 제거한다.

-c 와 -d 가 둘 다 결정되면 string1 에서 규정된것들을 제외한 모든 기호들이 제거된다. 만일 -s 가 규정되지 않으면 string2 의 내용은 무시된다. 그러나 동일한 기호렬이 -d 및 -s 둘 다에 대하여 리용될수는 없다. 즉 이 두개의 선택항목들이 규정되였을 때에는 string1 과 string2 가 둘 다 필요하다.

만일 -d 나 -s 가 없으면 매개 기호 혹은 조사성분들은 string2 에 의하여 규정된 동일한 위치의 기호 혹은 조사성분에 의하여 교체된다.

-s　string1 에 있는 어떤 기호가 두번 이상 반복되는 기호들의 렬로써 발생될 때 string2 안에 있는 대응하는 한개 기호로 교체된다.

만일 string2 가 기호모임을 포함한다면 그것은 그 기호모임에 속하는 모든 기호들을 포함하는것으로 인식된다.

실례:

```
tr -s '[:space:]'
tr -s '[:upper:]' '[:lower:]'
```

다음의 형식들은 기호구역을 표시하는데 리용된다.

c1-c2 혹은 [c1-c2]

조사성분 c1 부터 c2 까지의 구역으로서 LC_COLLATE 지역분류의 현재설정값으로 정의된다.

[:class:] 혹은 [[:class:]]

LC_CTYPE 지역 분류의 현재 설정값으로 정의되는 기호클라스에 속하는 모든 기호들을 의미한다. alnum, alpha, blank, cntrl, digit, graph, lower, punct, space, upper, xdigit 등은 string1 에서 규정되는 클라스이름들이다.

선택항목 -d 와 -s 가 함께 있을 때 임의의 기호클라스가 string2 로 될수 있다. 그밖의 경우에는 lower 혹은 upper 만이 string2 로 될수 있으며 또 그때에만 대응하는 기호클라스가 string1 안의 동일한 위치에서 규정된다.

[:lower:]이 string1 에서 나타나고 [:upper:]이 string2 에서 나타날 때 조사하려는 렬은 현재 지역의 LC_CTYPE 분류에 대응하는 toupper 의 기호들을 포함한다.

[:upper:]가 string1 에서 나타나고 [:lower:]가 string2 에서 나타날 때 조사하려는 렬은 현재 지역의 LC_CTYPE 분류에 대응하는 tolower 의 기호들을 포함한다.

[=c=] 혹은 [[=c=]]

c 와 동등한 클라스에 속하는 모든 기호 혹은 조사성분들을 의미한다. 이 클라스는 string1 에만 허용되며 -d 와 -s 가 있을 때에는 string2 에도 허용된다.

[a*n] a 의 n 번 반복을 의미한다. n 이 0 이거나 없으면 string1 의 길이에 따라 string2 에서 반복된다.

탈퇴문자 \ 은 기호렬안에 있는 임의의 기호를 제거하는 특수한 의미로 쉘에서 리용될수 있다. 또한 \ 뒤에 붙은 1, 2, 3 개의 8 진수자들은 그 기호의 ASCII 코드값이다.

string1 혹은 string2 에 있는 ASCII 의 NUL 기호는 탈퇴기호로서만 표시될수 있다.

외부적영향

환경변수들

LANG 은 설정되지 않았거나 비여 있는 국제화변수에 대한 기정값을 제공한다. LANG 이 결정되지 않았거나 null 이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 tr 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 은 비지 않은 기호렬값으로 설정되여 있으면 기타 국제화변수들의 모든 값을 무시하기 위한 지역을 결정한다.

LC_CTYPE 는 본문의 단일/다중바이트기호들, 인쇄가능한 기호들의 분류, 식에서 기호모임식들에 대한 해석을 결정한다.

LC_MESSAGES 는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용의 위치와 표준적인 출력장치에 씌여 지는 비형식적인 통보의 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 장소를 결정한다.

귀환값

0	성과적으로 완성
>0	오유발생

실례

ASCII 기호모임과 표준적인 조사렬에 대하여 file1 안에 있는 모든 단어들을 file2 에서 한 행에 한 단어씩 놓도록 하려면 다음과 같이 한다. 이때 특수기호들이 쉘에 의하여 해석되는것을 막기 위하여 기호렬을 " "안에 포함시킨다.(012 는 행바꾸기기호에 대한 ASCII 코드값이다.)

```
tr -cs "[A-Z] [a-z]" "[\012*]" <file1 >file2
```

모든 기호모임과 조사하려는 렬에 대하여 우의것과 동일하게 하려면 다음과 같이 하여야 한다.

```
tr -cs "[:alpha:]" "[\012*]" <file1 >file2
```

file1 의 모든 소문자들을 대문자로 변환하여 결과를 표준출력장치에 쓰려면 다음과 같이 하여야 한다.

```
tr "[:lower:]" "[:upper:]" <file1
```

동등클라스를 리용하여 file1 안의 기호 e 의 변종들을 식별하고 그것들의 식별표식을 없앤 다음 결과를 file2 에 쓰려면 다음과 같이 하여야 한다.

```
tr "[=e=]" "[e*]" <file1 >file2
```

file1 안에 있는 매개 수자들을 #으로 변환하고 결과를 file2 에 쓰려면 다음과 같이 하여야 한다.

```
tr "0-9" "[#*]" <file1 >file2
```

*는 첫번째 기호렬의 길이만큼 두번째 기호렬에서 #를 반복하여 표시하도록

한다.

저자

tr는 OSF와 HP에 의하여 개발되였다.

관련항목

ed(1), sh(1), ASCII(5), envion(5), lang(5), regexp(5)

표준일치

tr: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

wc

wc – 단어, 바이트, 행들을 계수한다.

---

wc(1)

이름

wc – 단어, 행, 바이트 혹은 기호들을 계수한다.

형식

wc [-c | -m] [-lw] [names]

해설

wc는 주어 진 파일 혹은 표준입력장치에서 행, 단어, 바이트 또는 문자들을 계수한다. 또한 주어 진 파일들에서 이것들의 총 개수도 계산한다.

단어는 공백, 태브, 행바꾸기기호에 의하여 구별되는 문자들의 최대렬이다.

선택항목들

-c 매개 입력파일안의 바이트수를 표준출력장치에 현시한다.

-m 매개 입력파일안의 기호개수를 표준출력장치에 현시한다.

-w 매개 입력파일안의 단어개수를 표준출력장치에 현시한다.

-l 매개 입력파일안의 행바꾸기기호개수를 표준출력장치에 현시한다.

c와 m들은 서로 배타적이다. 아무런 선택항목도 없으면 기정출력은 -lwc이다.

외부적영향

환경변수들

LC_CTYPE 는 도형과 공백기호들의 구역, 본문에 대한 단일/다중바이트기호들의 해석을 결정한다.

LC_MESSAGES 는 현시되는 통보문의 언어를 결정한다.

LC_CTYPE 나 LC_MESSAGES 가 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정값으로 쓰인다. LANG 이 결정되지 않았거나 null 이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 wc 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

경고

wc 는 행개수를 결정하기 위하여 행바꾸기기호개수를 계산하기때문에 본문파일의 마지막행이 행바꾸기기호가 없이 끝나면 개수는 1 만큼 작아 진다.

XPG 에서만은 표준적으로 매 입력파일에 대하여 표준출력장치에 다음과 같은 형식으로 출력된다.

"%d %d %d %s\n", <newlines> <words> <bytes> <file>

귀환값 0 성과적으로 완성

>0 오유발생

실례 : file1 에 있는 단어개수와 기호개수를 출력하려면 다음과 같이 하여야 한다.

-wc -wm file1

우의 지령은 다음과 같은 형식으로 출력된다.

n1 n2 file1

여기서 n1 은 단어개수이고 n2 는 기호개수이다.

표준일치

wc: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제6장. 개선된 UNIX도구 - 정규식, sed, awk, grep지령

이 장에서는 정규식에서 리용되는 sed, awk, grep지령들에 대하여 고찰한다. 여기서 awk는 sed와 grep에서 파생된것이며 이 세개의 지령들에서 리용되는 정규식들은 류사하다. 먼저 정규식에 대하여 고찰한 다음 이 세개의 지령에 대하여 고찰한다.

정 규 식

정규식(Regular Expression)은 사용자들이 탐색하는 패턴들을 서술한 기호렬식이다. 정규식에서는 보통 통용기호를 리용한다.

정규식은 쉘에서 리용되는 파일정합패턴들과는 다르다. 여기서 고찰하는 정규식은 쉘 및 다른 많은 프로그람들에서 다 리용되는 일반적인것이지만 파일정합패턴은 쉘과 find와 같은 프로그람에서만 리용되는 국부적인것이다.

정규식은 웃반점(')안에 써넣어 쉘의 인수로 리용할수 있다. 웃반점안에 포함된 기호들은 탐색을 조종하는 기호 즉 메타기호(메타문자)로 된다.

기호렬 및 통용기호의 사용

grep 혹은 vi와 같은 프로그람을 사용할 때 사용자가 정규식을 주면 그 프로그람은 그것을 탐색한다. 이때의 정규식은 하나의 기호렬이거나 통용기호일수 있다. 이러한 통용기호들을 여러 프로그람들에서 **메타기호**(Meta Character)라고 부른다.

표 6-1은 메타기호와 그것을 리용하는 프로그람들에 대한 목록을 주고 있다.

표 6-1 메타기호와 그 리용프로그람

메타기호	awk	grep	sed	vi	사 용
.	○	○	○	○	임의의 한 기호를 탐색한다.
*	○	○	○	○	*앞에 있는 기호가 임의의 개수만큼 반복되여 있을 때 그 기호렬을 탐색한다.
[…]	○	○	○	○	[…]안에 있는 임의의 한 기호를 탐색한다.
$	○	○	○	○	행의 끝을 탐색한다.
^	○	○	○	○	행의 시작을 탐색한다.
\	○	○	○	○	\뒤에 있는 특수기호를 탈퇴시킨다.

(표계속)

메타기호	awk	grep	sed	vi	사 용
\{n, m}\	○	○	×	×	n과 m사이에 있는 한 기호의 발생령역을 탐색한다.
+	○	×	×	×	이전의 정규식을 한번이상 탐색한다.
?	○	×	×	×	이전의 정규식을 기껏 한번 탐색한다.
|	○	×	×	×	이전 혹은 이후의 정규식을 탐색할수 있다.
( )	○	×	×	×	정규식을 표준적인 괄호형식으로 그룹화한다.
\{\}	×	×	×	○	단어의 시작 혹은 끝을 탐색한다.

sed

UNIX체계에서 편집작업은 대체로 vi에서 수행된다. 그러나 vi만 가지고서는 만족스럽게 파일편집을 할수 없다. 사용자가 쉘프로그람을 쓰거나 과제들사이의 정보를 결합시킬 필요가 있을 때 이런 편집작업은 체계와의 호상작용이 없이 진행되여야 한다. 바로 이러한 요구를 sed로 실현할수 있다. sed는 **흐름식편집기**(Stream Editor)라는 의미를 가지고 있다.

사용자는 편집하려는 파일이름을 sed에 규정할수 있으며 sed는 표준입력장치로부터 입력자료를 받아 들인다. sed는 한번에 한개 행씩 읽으며 매행에서 사용자가 규정하는 편집작업을 수행한다. 사용자는 sed에서 편집하려는 행번호들을 규정할수 있다.

sed의 지령들은 대부분 ed의 지령들과 같다. sed에는 다음과 같은 두가지 방법으로 입장할수 있다.

```
sed [-n] [-e] 'command' filename(s)

sed [-n] -f scriptfile filename(s)
```

첫번째 형식은 지령행에서 리용되는것이다. 표준적으로 sed는 모든 행들을 현시한다. 선택항목 -n을 규정하면 지령 p에 의하여 규정된 행들만 출력한다. -e는 지령행에서 한개이상의 지령들이 사용될 때 다음번 인수가 지령이라는것을 sed에게 통보하여 준다.

두번째 형식은 편집지령들을 포함하는 한개이상의 스크립트를 규정하도록 허용한다. 이 두 형식에서 리용되는 인수들은 다음과 같다.

-n　　　　　　지령 p에 의하여 규정되는 행들만 출력한다.

-e command　　-e다음에 오는 인수는 편집지령이다.

-f filename　　-f뒤에 오는 인수는 편집지령들을 포함하는 파일이다.

다음의 지령에서는 passwd.test파일을 지령 cat로 보고 있으며 그다음 sed의 p를 리용하여 16, 17, 18번 행들만 보고 있다.

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail;/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed -n  16, 18p passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
```

mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
col:WhOyzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
sed -n 16, 18p passwd.test
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux UBer:/home/col:/bin/bash

16, 17, 18번 행들만을 출력하려는 첫번째 시도에서는 16, 17, 18행들이 두번씩 출력되였다. 그 리유는 sed가 매행을 읽고 그 행에 대하여 작용을 하기때문이다. 행에 직접 작용하도록 하려면 -n을 사용할수 있다. 이 선택항목을 쓰면 모든 행들이 표준출력장치로 가는것을 막을수 있다. 이런 방법으로 16, 17, 18번 행들만을 볼수 있다.

다음의 실례에서는 passwd.test라는 파일을 다시 찾아 보고 d를 리용하여 우와 같은 세개의 지령을 제거하고 있다.

cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2;2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:

games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp;*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
sed 16, 18d passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:

이 실례에서는 16부터 18사이의 행들을 제거하도록 하고 있다. 이렇게 어떤 령역이 아니라 하나의 행만을 제거하도록 할수 있다. 그리고 초기파일은 그대로 남겨 두고 결과가 표준출력장치에 출력된다.

파일에서 패턴을 탐색할수도 있으며 그 패턴을 포함하는 행들만 제거할수도 있다. 다음의 실례에서는 bash를 탐색하고 그것을 포함하는 행들을 제거하고 있다.

 # cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*: 3:4: adm: /var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt

mail:*:8:12:mail:/var/spool/mail:
newe:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP user:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:-:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
sed '/ bash/ d' passwd. test
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*zl4:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:-:65534:65534:Nobody:/:/bin/false

이 실례에서 보는바와 같이 passwd. test로부터 bash를 포함하는 두개의 root행과 call행이 제거되였다. 또한 정규식들과 실행지령을 웃반점안에 포함시킨것을 볼수 있다.

bash를 포함하는 행들을 내놓고 나머지 모든 행들을 제거하려면 d앞에 !를 삽입해야 한다.

cat passwd. test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:-:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:

```
lp:*:4:7:lp:/var/spool/lpd:
sync:*;5:0:sync:/sbin:/bin/sync
shutdown:*:6:11;shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed '/bash/ !d' pasowd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

파일의 끝에 세개의 행들을 추가하려면 다음의 실례에서와 같이 하면 된다.

```
# sed '$a\
> This is a backup of passwd file\
> for viewing purposes only\
> so do not modify' passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:-:1:l:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
```

```
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
This is a backup of passwd file
for viewing purposes only
so do not modify
```

이 실례에서는 거꿀빗선(\)이 여러번 리용되였다. 매개 \은 행바꾸기기호를 표시한다. $로 파일의 끝으로 가서 \으로 새행을 추가한 다음 목적하는 본문과 그 본문뒤에 행바꾸기기호를 추가한다. 그렇게 하면 파일끝에 행들을 추가할수 있다.

다음의 실례는 파일의 앞에 새행들을 추가하고 있다.

```
# sed 'li\
>This in a backup passwd file\
>for viewing purposes only\
>so do not modify\
>' passwd.test
This is a backup passwd file
for viewing purposes only
so do not modify
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

이 실례에서는 sed지령뒤에 하나의 웃반점을 사용하였고 또 다른 하나의 웃반점은 모든 정보를 규정한 다음 마지막행에서 주고 있다.

아래에서는 sed의 개요를 준다.

sed – 흐름식편집기

지령들

a	본문을 추가한다.
b	표식으로의 갈래를 조성한다.
c	행들을 본문으로 교체한다.
d	현재의 본문완충기를 제거한다.
D	현재의 본문완충기의 첫행을 제거한다.
g	취해 진 공간의 내용을 덧쓰기하여 붙인다.
G	취해 진 공간의 내용을 덧쓰지 않고 그 주소아래에 붙인다.
h	패턴공간을 취해 진 공간으로 복사한다.
H	패턴공간의 내용을 취해 진 공간에 추가한다.
i	본문을 삽입한다.
l	패턴공간의 내용을 렬거한다.
n	입력에서 다음행을 패턴공간안으로 읽어 들인다.
N	입력의 다음행을 패턴공간에 덧붙인다.
p	패턴공간을 인쇄한다.
P	패턴공간의 시작부터 행바꾸기기호까지 인쇄한다.
q	주소가 계수되였을 때 탈퇴한다.
r	파일에서 읽는다.
s	패턴들을 치환한다.
t	치환이 현재 패턴공간에 만들어 졌는가에 따라 갈래를 조성한다.
w	패턴공간의 내용을 규정된 파일에 추가한다.
x	취해 진 령역과 패턴공간의 내용을 호상 교체한다.
y	기호들을 변환한다.

awk

awk는 파일의 탐색, 수정 그리고 보고서작성 등에 다양하게 리용될수 있다. awk는 입력장치(표준입력장치 혹은 파일)의 행들에서 패턴을 탐색함으로써 이러한 과제들을 수행한다. 주어 진 패턴을 탐색하는 매개 행들에서 awk는 그 행을 위한 몇가지 복잡한 처리들도 진행하게 된다. 실지로 입력행들을 탐색하는 코드는 쉘스크립트와 C프로그람의 결합으로 되여 있다.

grep, cut, paste에 의한 복잡한 결합연산들로 진행되는 자료조종과제들이 awk에서는

대단히 쉽게 수행된다. 왜냐하면 awk가 프로그람작성언어이므로 수학적인 연산도 할수 있고 입력자료의 검사도 쉽게 할수 있기때문이다(쉘에서는 수학적연산을 충분히 하지 못한다.). 또한 awk는 류점수연산까지도 할수 있다(쉘은 옹근수와 기호렬만을 취급한다.).

awk프로그람의 기본형식은 다음과 같다.

```
awk '/pattern_to_match/ {prog to run}' input_file_names
```

우에서 본바와 같이 전체 프로그람은 웃반점(')안에 포함된다. 파일이름이 하나도 규정되지 않으면 awk는 표준입력장치로부터 읽는다.

pattern_to_match는 실지 정규식으로서 빗선(/)사이에 반드시 놓여야 한다. 그리고 실행프로그람은 awk코드로 씌여 진다. 이 코드는 C와 비슷하다. 프로그람은 입력장치의 행이 정규식 pattern_to_match를 탐색할 때마다 수행된다. 그러나 /pattern_to_match/이 {}안에 들어 있는 프로그람의 앞에 놓이지 않으면 그 프로그람은 입력장치의 모든 행들에 대하여 실행된다.

awk는 입력행에서 마당들과 함께 작용한다. 마당은 공백이나 다른 마당분리기호로 구별되는 단어들이다(awk에서는 공백이 표준적인 마당분리기호로 쓰인다.). -F를 사용하면 마당분리기호를 다시 규정할수 있다. awk패턴들에 있는 마당과 프로그람들은 기호 $와 그뒤에 놓이는 파일번호에 의하여 참조된다. 실례로 입력행의 두번째 마당은 $2이다. 쉘프로그람안에서 awk지령들을 사용하는 경우에 마당들($1, $2 등)은 쉘스크립트의 위치파라메터와 충돌하지 않는다. 왜냐하면 awk변수들은 웃반점안에 포함되여 있으며 쉘은 그것들을 무시하기때문이다.

다음의 실례에서 사용하는 파일 newfiles는 15일간이하로 체계에 존재하는 파일들의 목록을 포함하고 있다. 이 파일은 UNIX체계의 여러 측면들을 검사하는 체계조직검사프로그람의 부분으로서 생성되여 있는것이다. 아래에서는 newfiles의 내용을 보여 주고 있다.

```
# cat newfiles
PROG>>>>> report of files not older than 14 days by
find the file system is /
-rw-r--r--   1 root   root     567  Dec  7 07:16   ./etc/mnttab
-rw-r--r--   1 root   root   20713  Dec  7 07:18   ./etc/rc.log
-rw-r--r--   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/hparray.map
-rw-r--r--   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/hparray.devs
-rw-r--r--   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/hparray.luns
-rw-r--r--   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/hparray.addr
-r-s------   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/pscan.lock
-r-s------   1 root   root       0  Dec  7 07:17   ./etc/hpC2400/monitor.lock
-rw-r--r--   1 root   root   14299  Dec  7  7:17   ./etc/hpC2400/HPARRAY.INFO
-rw-r--r--   1  bin    bin    8553  Dec  7 07:02   ./etc/shutdownlog
-rw-r--r--   1 root   mail   32768  Dec  7 07:16   ./etc/mail/aliases.db
-rw-r--r--   1 root   mail      33  Dec  7 07:16   ./etc/mail/Bendmail.pid
```

```
-rw-r--r--    1  root        root        13    Dec   7 07:16    ./etc/opt/dce/boot_time
-rw-r--r--    1  root        root       720    Dec   7 13:34    ./etc/utmp
-rw-r--r--    1  root        root         0    Dec   7 07:16    ./etc/xtab
-rw-r--r--    1  root        root         0    Dec   7 07:18    ./etc/rmtab
-rw-r--r--    1  root        root     40814    Dec   7 07:15    ./etc/rc.log.old
-rw-r--r--    1  root        root      4620    Dec   7 13:34    ./etc/utmpx
-rw-r--r--    1  root        root         9    Dec   7 13:17    ./etc/ntp.drift
-rw-r--r--    1  root        root       616    Dec   7 07:15    ./etc/auto_parms.log
-rw-r--r--    1  root         sys       219    Dec   7 07:00    ./etc/auto_parms.log.old
-rw-rw-rw-    1  root         sys       520    Nov  23 12:37    ./.sw/sessions/swliBt.last
-r--r--r--    1  root    informix        76    Dec   7 07:17    ./INFORMIXTMP/.inf.shmPSREP
-r--r--r--    1  root    informix        76    Dec   7 07:18    ./INFORMIXTMP/.inf.shmPSDEV
-rw-------    1autosys autosys         4052    Nov  25 14:08    ./home/autosys/.sh_history
-rw-------    1tsaxs        users      2228    Dec   1 13:15    ./home/tsaxs/.sh_history
-rw-------    1tsfxo        users      2862    Nov  24 10:08    ./home/tsfxo/.sh_history
PROG>>>>> report of files not older than 14 days by find
the file system is /usr
-rw-rw-rw-    1opop6       users        21    Dec   7 13:46    ./local/adm/etc/lmonitor.hst
-rw-r--r--    1 tsgjf      users      1093    Dec   7 13:17    ./local/flexlm/licenses/license.log
PROG>>>>> report of files not older than 14 days by find
the file system is /opt
-rw-rw-r--    1   bin         bin       200    Dec   7 07:17    ./pred/bin/OPSDBPF
-rw-r--r--    1  root         sys    800028    Dec   7 07:17    ./pred/bin/PSRNLOGD
PROG>>>>> report of files not older than 14 days by find
the file system is /var
-rw-r--r--    1  root         sys     45089    Dec   7 07:16    ./adm/sw/swagentd.log
-rw-rw-rw-    1  root         sys       562    Dec   7 07:16    ./adm/sw/sessions/swlist.last
-rw-rw-r--    1  root        root     12236    Dec   7 07:16    ./adm/ps_data
-rw-r--r--    1  root        root        65    Dec   7 07:17    ./adm/cron/log
-rw-r--r--    1  root        root       162    Dec   7 07:00    ./adm/cron/OLDlog
-r--r--r--    1  root        root    734143    Dec   7 07:16    ./adm/syslog/mail.log
-rw-r--r--    1  root        root     65743    Dec   7 13:56    ./adm/sysiog/sysiog.log
-rw-r--r--    1  root        root   4924974    Dec   7 07:02    ./adm/syslog/OLDsyslog.log
-rw-rw-r--    1  adm          adm   2750700    Dec   7 13:52    ./adm/wtmp
-rw-------    1  root       other    145920    Dec   3 14:36    ./adm/btmp
-rw-r--r--    1    lp          lp        33    Dec   7 07:17    ./adm/lp/log
-rw-r--r--    1    lp          lp        67    Dec   7 07:01    ./adm/lp/oldlog
-rw-r--r--    1  root        root      4330    Dec   7 07:18    ./adm/diag/device_table
-rw-r--r--    1  root        root        34    Dec   7 07:18    ./adm/diag/misc_sys_data
```

```
-rwxr-xr-x  1 root    root  995368  Nov 22 15：16   . /adm/diag/LOG0190
-rwxr-xr-x  1 root    root  995368  Nov 23 02：05   . /adm/diag/LOG0191
-rwxr-xr-x  1 root    root  453964  Nov 23 07：01   . /adm/diag/LOG0192
-rwxr-xr-x  1 root    root  970448  Nov 23 18：35   . /adm/diag/LoGO193
-rwxr-xr-x  1 root    root  995368  Nov 24 05：24   . /adm/diag/LoGO194
-rwxr-xr-x  1 root    root  995368  Nov 24 16：14   . /adm/diag/LoGO195
-rwxr-xr-x  1 root    root  995368  Nov 25 03：03   . /adm/diag/LOG0196
-rwxr-xr-x  1 root    root  995368  Nov 25 13：52   . /adm/diag/LOG0197
-rwxr-xr-x  1 root    root  995368  Nov 26 00：41   . /adm/diag/LOG0198
-rwxr-xr-x  1 root    root  995368  Nov 26 11：31   . /adm/diag/LOG0199
-rwxr-xr-x  1 root    root  995368  Nov 26 22：20   . /adm/diag/LOG0200
-rwxr-xr-x  1 root    root  995368  Nov 27 09：09   . /adm/diag/LoG0201
-rwxr-xr-x  1 root    root  995368  Nov 27 19：58   . /adm/diag/LoG0202
-rwxr-xr-x  1 root    root  995368  Nov 28 06：48   . /adm/diag/LOG0203
-rwxr-xr-x  1 root    root  995368  Nov 28 17：37   . /adm/diag/LoG0204
-rwxr-xr-x  1 root    root  995368  Nov 29 04：26   . /adm/diag/LOG6205
-rwxr-xr-x  1 root    root  995368  Nov 29 15：16   . /adm/diag/LOG0206
-rwxr-xr-x  1 root    root  995368  Nov 30 02：05   . /adm/diag/LoG0207
-rwxr-xr-x  1 root    root  452020  Nov 30 06：59   . /adm/diag/LoG0208
-rwxr-xr-x  1 root    root  970448  Nov 30 18：35   . /adm/diag/LoG0209
-rwxr-xr-x  1 root    root  995368   Dec  1 05：24   . /adm/diag/LOG0210
-rwxr-xr-x  1 root    root  995368   Dec  1 16：13   . /adm/diag/LOG0211
-rwxr-xr-x  1 root    root  995368   Dec  2 03：03   . /adm/diag/LOG0212
-rwxr-xr-x  1 root    root  995368   Dec  2 13：52   . /adm/diag/LoG0213
-rwxr-xr-x  1 root    root  995368   Dec  3 00：41   . /adm/diag/LOG0214
-rwxr-xr-x  1 root    root  995368   Dec  3 11：31   . /adm/diag/LOG0215
-rwxr-xr-x  1 root    root  995368   Dec  3 22：20   . /adm/diag/LOG0216
-rwxr-xr-x  1 root    root  995368   Dec  4 09：09   . /adm/diag/LOG0217
-rwxr-xr-x  1 root    root  995368   Dec  4 19：58   . /adm/diag/LOG0218
-rwxr-xr-x  1 root    root  995368   Dec  5 06：48   . /adm/diag/LoG0219
-rwxr-xr-x  1 root    root  995368   Dec  5 17：37   . /adm/diag/LoG0220
-rwxr-xr-x  1 root    root  995368   Dec  6 04：26   . /adm/diag/LOG0221
-rwxr-xr-x  1 root    root  995368   Dec  6 15：15   . /adm/diag/LCG0222
-rwxr-xr-x  1 root    root  995368   Dec  7 02：05   . /adm/diag/LOG0223
-rwxr-xr-x  1 root    root  453964   Dec  7 07：00   . /adm/diag/LOG0224
-rwxr-xr-x  1 root    root  543740   Dec  7 13：57   . /adm/diag/LoG0225
-rw-r--r--  1 root    root   19587   Dec  7 07：16   . /adm/ptydaemonlog
-rw-r--r--  1 root    root      52   Dec  7 07：16   . /adm/conBlOg. OptS
-rw-r--r--  1 root    root       0   Dec  7 07：16   . /adm/rpc. statd. log
```

```
-rw-r--r--    1 root      root         0  Dec  7 07:16  ./adm/rpc.lockd.log
-rw-r--r--    1 root      root     24250  Dec  7 07:16  ./adm/vtdaemonlog
-rw-------    1 root      root       214  Dec  7 12:07  ./adm/sulog
-rw-------    1 root      root       381  Dec  3 17:34  ./adm/OLDsulog
-rw-r--r--    1 root      sys        145  Dec  7 07:16  ./adm/rbootd.log
-rw-------    1sysadm     psoft       60  Dec  1 16:59  ./tmp/EAAa09O57
-rw-r--r--    1 tsgjf     users        0  Dec  7 13:17  ./tmp/lockHPCUPLANGS
-rw-r--r--    1 tsgjf     users      175  Dec  7 06:40  ./tmp/.flexlm/lmgrd.1507
-rw-r--r--    1 tsgjf     users      175  Dec  7 13:28  ./tmp/.flexlm/lmgrd.1505
-rw-r--r--    1   lp      lp           0  Dec  7 07:17  ./spool/lp/outputq
-rw-rw-rw-    1   lp      lp           4  Dec  7 07:17  ./spool/lp/SCHEDLOCK
-rw-------    1 root      sys          0  Nov 23 07:00  ./spool/cron/tmp/croutAAAa01030
-rw-------    1 root      sys          0  Nov 30 07:00  ./spool/cron/tmp/croutAAAa01039
-rw-------    1 root      sys          0  Dec  7 07:00  ./spool/cron/tmp/croutAAAb01039
-rw-r--r--    1 root      root         4  Dec  7 07:16  ./run/syslog.pid
-rw-r--r--    1 root      root         4  Dec  7 07:16  ./run/gated.pid
-rw-r--r--    1 root      sys        145  Dec  7 07:16  ./run/gated.version
-rw-r--r--    1 root      sys          3  Dec  7 07:16  ./statmon/state
-rw-r--r--    1 root      root     29771  Dec  7 07:16  ./opt/dce/config/dce_config.log
-rw-r--r--    1 root      sys         74  Dec  7 07:16  ./opt/dce/rpc/local/00404/srvr_socks
-rw-r--r--    1 root      root        72  Dec  7 07:16  ./opt/dce/rpc/local/00927/srvr_socks
-rw-r--r--    1 root      root     32768  Dec  7 07:16  ./opt/dce/dced/Ep.db
-rw-r--r--    1 root      root     32768  Dec  7 07:20  ./opt/dce/dced/Llb.db
-rw-r--r--    1 root      root         0  Nov 30 07:16  ./opt/perf/status.ttd
-rw-r--r--    1 root      root        33  Dec  7 07:17  ./opt/perf/datafiles/RUN
-rwxrwxrwx 1 root      sys    9243180  Dec  7 13:55  ./opt/perf/datafiles/logappl
-rwxrwxrwx 1 root      sys    8697612  Dec  7 13:55  ./opt/perf/datafiles/logdev
-rwxrwxrwx 1 root      sys    9195152  Dec  7 13:55  ./opt/perf/datafiles/logglob
-rwxrwxrwx 1 root      sys      11112  Dec  7 07:17  ./opt/perf/datafiles/logindx
-rwxrwxrwx 1 root      sys   17639080 Dec  7 13:57  ./opt/perf/datafiles/logproc
-rwxrwxrwx 1 root      sys       3797  Dec  7 07:17  ./opt/perf/datafiles/mikslp.data
-rw-rw-rw-    1 root      sys        105  Nov 30 10:45  ./opt/perf/datafiles/agdb
-rw-r--r--    1 root      root         5  Dec  7 07:17  ./opt/perf/datafiles/.perflbd.pid
-rw-rw-rw-    1 root      sys      21176  Dec  7 07:20  ./opt/perf/status.scope
-rw-rw-rw-    1 root      root         5  Nov 30 07:16  ./opt/perf/ttd.pid
-rw-r--r--    1 root      root         0  Dec  7 07:17  ./opt/perf/status.mi
-rw-rw-rw-    1 root      sys       8254  Dec  7 07:17  ./opt/perf/status.perf1bd
-rw-rw-rw-    1 root      sys      21507  Dec  7 07:20  ./opt/perf/status.rep_server
-rw-rw-rw-    1 root      sys      24570  Dec  7 07:20  ./opt/perf/status.alarmgen
```

```
-rw-rw-rw-  1  root         sys  160956    Dec   6 21:13     ./opt/omni/log/inet.log
-rw-rw-rw-  1  root         sys  158796    Dec   7 07:17     ./sam/iog/samiog
-rw-r--r--  1  root        root   64730    Dec   7 07:17     ./sam/boot.config
-rw-rw-rw-  1  root         sys   11906   Nov  24 14:27     ./sam/poe.iout
-rw-rw-rw-  1  root         sys   11906   Nov  23 09:10     ./sam/poe.iout.old
-rw-rw-rw-  1  root         sys       29   Nov  24 14:27     ./sam/poe.dion
```

이 실례에서 보는것처럼 이 파일은 공백으로 구분되는 여러개의 마당들을 포함하고 있다.

다음의 실례에서는 세번째 마당이 "adm"과 꼭 같은가를 평가하고 그런 행을 출력한다.

```
# awk ' $3  = =" adm" { print }' newfiles
-rw-rw-r--   1 adm   adm     2750700  Dec    7  13:52  ./adm/wtmp
```

우에서 보는것처럼 세번째 마당의 "adm"만을 포함하는 행은 하나이다.

다음의 실례는 세번째 마당이 근사적으로 "adm"과 같은가를 평가하고 있다. 근사적으로 동일하다는것은 세번째 마당안에 "adm"을 포함하고 있다는것을 의미한다.

```
# awk '$3 ~ " adm "  {print}' newfiles
-rw-rw-r--   1     adm    adm   2750700  Dec   7 13:52    ./adm/wtmp
-rw-------    1  sysadm   psoft          60  Dec   1 16:59    ./tmp/EAAa09057
```

이 실례에서 출력된 마지막행은 세번째 마당이 "sysadm"으로서 그 마당안에 "adm"이 들어 있다.

다음의 실례는 앞의 실례와 거의 같지만 9번째 및 5번째 마당만이 현시된다.

```
# awk '$3 ~ "adm" {print $9,  $5}'  newfiles
./adm/wtmp  2750700
./tmp/EAAa09057  60
```

이 실례에서는 9번째 마당인 파일의 이름과 5번째 마당인 그 파일의 크기만이 출력된다.

다음의 실례에서는 세번째 마당이 "root"와 같지 않은가를 평가하고 같지 않을 때에는 행전부를 출력한다.

```
# awk,   "$3 ! = "root" {print}'   newfiles
PROG>>>>> report of files not older than 14 days by find
the file system is /
-rw-r--r--    1     bin        bin     8553    Dec      7 07:02     ./etc/shutdownlog
-rw-------    1autosys   autosys    4052    Nov    25 14:08     ./home/autosys/.sh_history
-rw-------    1   tsaxs       users     2228    Dec      1 13:15     ./home/tsaxs/.sh_history
-rw-------    1   tsfxo       users     2862    Nov    24 10:08     ./home/tsfxo/.sh-history
```

```
PROG>>>>> report of files not older than 14 days by find
the file system is /usr
-rw-rw-rw-  1 opop6    users     21   Dec   7 13:46   ./local/adm/etc/lmonitor.hst
-rw-r--r--   1  tsgjf    users   1093   Dec   7 13:17   ./local/flexlm/licenBes/license.log
PROG>>>>> report of files not older than 14 days by find
the file system is /opt
-rw-rw-r--   1   bin      bin     200   Dec   7 07:17   ./pred/bin/OPSDBPF
PROG>>>>> report of files not older than 14 days by find
the file system is /var
-rw-rw-r--   1   adm      adm2750700   Dec   7 13:52   ./adm/wtmp
-rw-r--r--   1    lp       lp      33   Dec   7 07:17   ./adm/lp/log
-rw-r--r--   1    lp       lp      67   Dec   7 07:01   ./adm/lp/oldlog
-rw -------   1sysadm psoft 60Dec1 16:59./tmp/EAAa09057
-rw-r--r--   1  tsgjf    users      0   Dec   7 13:17   ./tmp/lockHPCUPLANGS
-rw-r--r--   1  tsgjf    users    175   Dec   7 06:40   ./tmp/.flexlm/lmgrd.1507
-rw-r--r--   1  tsgjf    users    175   Dec   7 13:28   ./tmp/.flexlm/lmgrd.1505
-rw-r--r--   1    lp       lp       0   Dec   7 07:17   ./spool/lp/outputq
-rw-rw-rw-  1    lp       lp       4   Dec   7 07:17   ./spool/lp/SCHEDLOCK
```

이 실레에서 매 마당은 공백으로 구분되여 있는데 UNIX체계에서는 보통 두점(:)을 마당분리기호로 사용한다. 아래의 실례에서는 passwd.test파일을 보여 주고 있다.

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
```

nobody:*:65534:65534:Nobody:/:/bin/false

col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash

-F를 리용할 때 선택항목뒤에 마당분리기호를 규정하면 된다.

다음의 실례에서는 마당분리기호로 두점(:)을 규정하고 passwd.test파일에서 첫번째 마당이 "root"와 같은 행들만을 출력한다.

\# **awk -F: '$1 = = "root" {print}, passwd.test**

root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash

다음의 실례에서는 마당분리기호를 규정하고 네번째 마당이 "0"과 같은 행들만을 출력한다(이것은 사용자가 "root"그룹의 한 성원이라는것을 의미한다.).

 # awk -F: '$4 = = "0" {print}' passwd.test

root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash

sync:*:5:0:sync:/sbin:/bin/sync

halt:*:7:0:halt:/sbin:/sbin/halt

operator:*:11:0:operator:/root:

awk에서는 == 이외의 비교연산들도 수행할수 있다.

다음의 실례에서는 passwd.test파일에서 14보다 작은 값을 가지는 그룹의 성원들을 출력하고 있다.

 # awk -F: '$4 < 14 {print}' passwd.test

root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash

bin:*:1:1:bin:/bin:

daemon:*:2:2:daemon:/sbin:

adm:*:3:4:adm:/var/adm:

lp:*:4:7:lp:/var/spool/lpd:

sync:*:5:0:sync:/sbin:/bin/sync

shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown

halt:*:7:0:halt:/sbin:/sbin/halt

mail:*:8:12:mail:/var/spool/mail:

news:*:9:13:news:/var/spool/news:

operator:*:11:0:operator:/root:

다음의 실례에서는 passwd.test파일에서 14이하의 값을 가지는 그룹의 성원들을 출력하고 있다.

 # awk -F: '$4 < = 14 {print}' passwd.test

root:PgYQCkVH65hyQ:0:0:roat:/root:/bin/bash

bin:*:1:1:bin:/bin:

daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:

다음의 실례에서는 passwd.test파일에서 14가 아닌 값을 가지는 그룹의 성원들을 출력하고 있다.

 # awk -F: '$4 ! = 14 {print}' passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
operator:*:11:0:operator:/root:
games:*:l2:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash

다음의 실례에서는 passwd.test파일에서 14이상의 값을 가지는 그룹의 성원들을 출력하고 있다.

 # awk -F: '$4 >= 14 {print}' passwd.test
uucp:*:10:14:uucp:/var/spool/uucp:
games:*:12:100:games:/usr/games:

```
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:1S:Manuals owner:/:
nobody:*:GS534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

다음의 실례에서는 passwd.test파일에서 14보다 큰 값을 가지는 그룹의 성원들을 출력하고 있다.

```
# awk -F: '$4 > 14 {print}' passwd.test
games:*:12:100:games:/usr/games:
gopher:*:13:30;gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

awk의 보충적인 실례들은 쉘프로그람작성을 취급하는 장에서 더 고찰하기로 한다.
아래에서는 awk의 비교연산자들에 대한 개요를 주고 있다.

awk - 행에서 주어 진 패턴을 탐색하며 연산들을 수행한다.

비교연산자들:

<	보다 작다.
<=	같거나 작다.
==	같다.
~	기호렬 탐색.
!=	같지 않다.
>=	같거나 크다.
>	보다 크다.

grep

grep는 본문을 탐색하고 그것을 현시하는데 쓰인다. grep는 정규식을 표준적으로 해석하는 구문해석자이다. 그림 6-1은 /home/denise를 렬거하고 grep를 리용하여 패턴을 탐색하고 있다.

```
grep example
$ ls -a /home/denise | grep netscape
.netscape-bookmarks.html
.netscape-cache
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
$ ls -a /home/denise | grep -c netscape
6
$ ls -a /home/denise | grep NETSCAPE
$ ls -a /home/denise | grep -i NETSCAPE
.netscape-bookmarks.html
.netscape-cache
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
$ ls -a /home/denise | grep -F "netscape
> .c"
.cshrc
.cshrc.orig
.netscape-bookmarks.html
.netscape-cache
.netscape-history
.netscape-newsgroups-news.spry.com
.netscape-newsgroups-newsserv.hp.com
.netscape-preferences
.newsrc-news.spry.com
.newsrc-newsserv.hp.com
$ █
```

그림 6-1. 지령 grep

그림 6-1에서는 먼저 패턴 netscape를 탐색하고 .netscape로 시작하는 모든 행들을 현시하고 있다.

그다음 -c를 리용하여 netscape의 발견회수를 출력한다. 결과는 6이다.

다음은 -i를 리용하여 대소기호의 차이를 무시하고 netscape패턴을 다시 탐색한다. 결과는 초기의것과 같다.

한개이상의 패턴도 탐색할수 있다. -F를 리용하여 두개의 패턴 netscape와 .c를 둘다 탐색하면 그림에서와 같이 더 많은 결과행들이 현시된다. 이때 두개의 패턴은 " "안에 포함되여야 하며 그안에서 패턴들은 서로 다른 행에 놓여야 한다.

grep를 리용하여 보다 발전된 탐색들에 대하여 고찰해 보자. 우의 실례에서 고찰한 passwd.test파일은 매행에 많은 정보를 포함한다. 아래에서는 Linux체계에서 passwd.test파일의 내용을 현시하고 있다.

```
# cat passwd.test
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
```

shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash

grep의 실례에서 진행한 탐색을 암호파일의 기호렬에 대하여서도 진행할수 있다. 아래의 실례에서는 passwd.test파일에서 news를 탐색한다.

\# **grep news passwd.test**
news:*:9:13:news:/var/spool/news:

passwd.test파일에서 bin이라는 이름을 가진 사용자가 있는가를 검사하려면 우선 다음과 같이 한다.

\# **grep bin passwd.test**
root:PgYQCkVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash

기호렬bin을 포함하는 많은 행들이 생성되였는데 사용자bin을 찾으려면 매행에서 처음으로 나타나는 기호렬이 bin이여야 한다. 따라서 탐색을 보다 질적으로 하려면 특수기호(ˆ)를 리용하여 매행의 첫 시작만 탐색하도록 다음과 같이 할수 있다.

\# **grep ^bin passwd.test**
bin:*:1:1:bin:/bin:

이 탐색결과는 정확히 bin으로 시작하는 행만을 주고 있다. 아래의 실례에서 보는것처럼 특수기호는 웃반점(')안에 포함하는것이 좋다. 왜냐하면 이러한 특수기호들은 쉘에 의하여 지령의 인수로서 해석될수 있기때문이다. 웃반점(')안에 포함되여야 정규식이라는것을 정확히 담보해 줄수 있다.

```
# grep '^bin' passwd.test
bin:*:1:1:bin:/bin:
```

-n을 리용하면 탐색한 행의 행번호까지도 출력하게 할수 있다.

 # grep -n '^bin' passwd.test
2:bin:*:1:1:bin:/bin:

아래에서는 grep지령에 대한 개요를 주고 있다.

grep - 본문을 탐색하고 그 결과를 현시한다.

선택항목들

-c	본문을 현시함이 없이 탐색한 행의 개수를 귀환한다.
-h	파일이름에 대한 참조가 없이 본문을 현시한다.
-i	탐색할 때 대소기호차이를 무시한다.
-l	본문을 현시함이 없이 정규식을 포함하는 파일의 이름을 귀환한다.
-n	본문과 함께 파일에서 탐색된 본문의 행번호를 귀환한다.
-v	정규식을 포함하지 않는 행들을 귀환한다.
-E	한개이상의 패턴을 탐색한다(egrep와 같다.).
-F	한개이상의 패턴을 탐색한다(fgrep와 같다.).

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX변종들에서 지령들은 좀 차이나기때문에 지령들의 선택항목나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

awk

awk – 패턴을 처리하는 언어.

awk(1)

이름

awk – 패턴을 처리하는 언어.

형식

awk [-Ffs] [-v var = value] [program | -f progfile …] [file …]

해설

awk는 program 혹은 한개이상의 -f progfile에서 행별로 주어 진 패턴을 탐색한다. 패턴에는 탐색할 때 수행해야 할 일련의 작용들이 있을수 있다. 이러한 작용들은 탐색된 매개 패턴에 대하여 수행된다. 여기서 var=value는 대입명령문으로서 선택항목 -v가 없으면 그것이 파일이름일 때 열리게 된다.

입력행은 공백이나 정규식FS에 의하여 구별되는 마당들로 이루어 진다. 매 마당들은 $1, $2, … ; $0으로 참조된다.

선택항목들

-F fs 마당들을 분리하는 정규식을 규정한다. 표준적으로는 공백과 태브 등으로 마당들을 인식한다. -F가 있으면 마당분리기호가 규정된다.

-f progfile
awk프로그람파일을 규정한다. 100개까지 가능하다. 이 파일들안에 있는 패턴-작용명령문들은 파일이 규정된 순서로 수행된다.

-v var=value
BEGIN작용이 수행되기전에 var=value대입이 수행되도록 한다.

명령문들

패턴-작용명령문은 다음의 형식을 가진다.

pattern { action }

{ action }이 생략되면 행들을 출력한다는것을 의미한다. pattern이 생략되면 항상 탐색을 한다는것을 의미한다. 패턴-작용명령문들은 행바꾸기기호 혹은 반두점(;)에 의하여 구별된다.

action은 명령문들의 렬이다.
명령문들은 다음과 같다.

```
if(식) 명령문 [else 명령문]
while(식) 명령문
for(식;식;식) 명령문
for(변수 in 배렬) 명령문
do 명령문 while(식)
break
continue
{[명령문 …]}
식              #보통 변수=식
print [식목록] [>식]
printf format [식목록] [>식]
return [식]
next          #나머지 패턴에로 뛰여 넘는다.
delete 배렬 [식]        #한개의 배렬원소를 제거한다.
exit [식]      #즉시 탈퇴한다; 이때의 상태는 식에 반영된다.
```

명령문들은 반두점, 행바꾸기기호, 오른쪽 괄호 등으로 끝난다. 빈 식목록은 $0으로 된다. 기호렬상수들은 " "안에 포함된다. 식들은 기호렬이나 수값을 취하며 +, -, *, /, %, ^연산자들에 의하여 작성된다. 연산자들로서는 ++, --, +=, -=, *=, /=, %=, ^=, **=, >, >=, <, <=, ==, !=, ?:들이 될수 있다. 변수들은 스칼라값, 배렬원소(x[i]) 등이 될수 있다. 변수는 빈 기호렬로 초기화된다. 배렬의 첨수는 수값만이 아니라 임의의 기호렬도 될수 있다. [i, j, k]와 같은 다중첨수도 될수 있다. print명령문은 표준출력장치(혹은 >file이거나 >>file이 존재하는 경우에는 파일, |이 존재하는 경우에는 파이프)에 인수들을 출력한다. printf명령문은 format에 따라 인수들을 형식화하여 출력한다.

출력함수들

전형적인 함수들인 exp, log. sqrt, sin. cos, atn2외에 다음과 같은 체계함수들이 더 제공된다.

blenth[([s])]
: 주어 진 기호렬인수의 바이트수를 귀환한다. 인수가 없으면 $0을 귀환한다.

lenth[([s])]
: 주어 진 기호렬인수의 기호의 개수를 귀환한다. 인수가 없으면 $0을 귀환한다.

rand( )　　0과 1사이의 하나의 우연수를 귀환한다.

srand([expr])

우연수발생을 위한 중심값을 설정한다. 인수가 주어 지지 않으면 현재의 시간이 중심값으로 리용된다.

int(x)　　옹근수값으로 자른다.

substr(s, m[, n])

기호렬 s에서 m번째 위치로부터 n개의 기호를 귀환한다. n이 생략되면 m번째 위치부터 마지막까지 잘라 낸 기호렬을 귀환한다.

index(s, t)

기호렬 s에서 1번부터 기호단위로 순서화하여 t가 처음으로 나타나는 위치를 귀환한다.

match(s, ere)

기호렬 s에서 1번부터 기호단위로 순서화하여 확장된 정규식 ere가 나타나는 위치를 귀환한다. 이때 RSTART와 RLENGTH 변수들은 탐색된 기호렬의 위치와 길이값으로 설정된다.

split(s, a[, fs])

기호렬 s를 배렬원소 a[1], a[2], …, a[n]으로 갈라 놓고 n값을 귀환한다. 분리는 정규식 fs에 의하여 진행되거나 그것이 없으면 마당분리기호 FS로 분리한다.

sub(ere, repl [, in])

기호렬 in에서 처음으로 나타나는 확장된 정규식ere를 repl로 귀환한다. in이 주어 지지 않으면 $0이 귀환된다.

gsub

정규식의 모든 발생이 교체된다는것을 제외하고는 sub와 같다. sub와 gsub는 교체회수를 귀환한다.

sprintf(fmt, expr, …)

fmt형식에 따라 expr기호렬을 형식화하여 출력한다.

system(cmd)

cmd를 수행하고 그것의 탈퇴상태를 귀환한다.

toupper(s)

기호렬 s를 대기호로 변환하고 그 결과를 귀환한다.

tolower(s)

기호렬 s를 소기호로 변환하고 그 결과를 귀환한다.

체계함수 getline은 현재 입력파일로부터 다음번 입력기록에 $0을 설정한다. 즉 getline < file은 file로부터 다음번 입력기록에 $0을 설정한다. getline x는 변수 x를 설정한다. cmd |getline는 cmd의 결과를 getline에 련결시킨다. 이 모든 경우에 getline는 성과적으로 입력하면 1을 귀환하고 파일끝에서 0, 오유가 있으면 -1을 귀환한다.

패턴

패턴들은 정규식과 관계식들을 임의로 론리적결합(!, ||, &&)한것이다. awk는 regexp(5)에서 서술된 확장된 정규식을 지원한다. 절대적인 정규식은 전체 행에 적용된다. 정규식은 연산자 ~ 와 !~ 를 리용하는 관계식에서 나타날수도 있다. /re/는 상수적인 정규식이다. 임의의 기호렬(상수 혹은 변수)은 정규식으로 리용될 수 있다.

한개의 패턴은 반점(,)으로 구분된 두개의 패턴들로 구성될수 있으며 이 경우에 패턴-작용은 두개의 패턴의 발생모두에 대하여 수행된다.

관계식들은 다음과 같다.

식 탐색연산자 정규식
식 관계연산자 식
식 in 배렬이름
(식, 식, …) in 배렬이름

여기서 관계연산자는 C언어에 있는 6개의 관계연산자와 같고 탐색연산자는 ~ 및 !~이다. 식은 산수식, 관계식, 두 식의 론리적결합으로 될수 있다.
특수패턴 BEGIN과 END는 첫 입력행을 읽기전과 마지막행을 읽은후에 조종을 받기 위하여 리용된다. BEGIN과 END는 다른 패턴들과 결합되지 않는다.

특수기호들

\ a	임의의 기호
\ b	<Backspace>기호
\ f	<form-feed>기호
\ n	행바꾸기기호
\ r	행끌기호
\ t	태브기호
\ v	수직태브기호
\ nnn	1~3개 수자로 된 8진수
\ xhhh	1~n개 수자로 된 16진수

변수이름들

FS 입력마당분리기호의 패턴

NF 현재 기록의 마당개수

NR 입력장치의 시작부터 현재 기록의 초기번호. BEGIN작용안에서 그 값은 0이다. END작용안에서 그 값은 처리된 마지막기록의 번호이다.

FNR 현재 파일의 현재 기록의 초기번호. BEGIN작용안에서 그 값은 0이다. END작용안에서 그 값은 마지막으로 처리된 파일의 마지막기록의 번호이다.

FILENAME
현재 입력파일의 경로이름.

RS 입력기록의 분리기호
표준적으로는 행바꾸기기호이다.

OFS print명령문의 출력마당분리기호
표준적으로는 행바꾸기기호이다.

ORS print명령문의 출력기록분리기호
표준적으로는 행바꾸기기호이다.

OFMT 수값에 대한 출력형식
표준적으로는 %.6g이다.

CONVFMT
수값에 대한 내부적인 변환형식
표준적으로는 %.6g이다.

SUBXEP 다차원배렬에 대한 첨수분리기호
표준값은 "34"이다.

ARGC ARGV안의 원소개수

ARGV 지령행인수들의 배렬. 선택항목과 0부터 ARGC-1까지 순서화된 프로그람인수의 개수들도 포함된다.

ENVIRON
환경변수들의 배렬
첨수들은 이름들이다.

RSTART
탐색된 기호렬의 시작위치
이 값은 항상 match함수의 귀환값과 같다.

RLENTH

탐색된 기호렬의 길이

함수들은 패턴-작용명령문의 위치에서 다음과 같이 정의된다.

function foo（a，b，c）{ …，return x }

파라메터들은 스칼라값，참조값(배렬이름인 경우)이 될수 있다. 함수들은 재귀적으로도 호출할수 있다. 파라메터들은 그 함수에 국부화되며 기타 변수들은 대역화된다.

패턴-작용명령문들이 HP-UX지령행에서 awk지령에 대한 인수로 리용될 때 그 명령문들은 웃반점(’)안에 포함되여야 한다.
실례로 길이가 72보다 큰 기호렬을 출력하라는 패턴-작용명령문은 -f progfile지령형식에서 다음과 같다.

length > 72

동일한 패턴-작용명령문을 awk지령의 인수로 리용할 때에는 다음의 형식으로 쓴다.

awk 'length >72'

외부적영향

환경변수들

LANG LANG은 설정되지 않았거나 비여 있는 국제화변수들에 대한 기정값을 제공한다. LANG이 설정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 변수가 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모든 값을 무시한다.

LC_CTYPE

본문의 단일/다중바이트기호들，인쇄가능한 기호들의 분류，식에서 기호모임식들에 대한 해석을 결정한다.

LC_NUMERIC

수값입력을 해석할 때 리용하는 밀수기호를 리용된다. 지역에 관계없이 .은 소수점으로 인식된다.

LC_COLLATES

정규식에서 령역의 작용，동등클라스들과 다중기호코드모임을 위한 지역을 결정한다.

LC_MESSAGES

표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용의 위치와 표준적인 출력장치에 씌여 지는 비형식적인 통보에 영향을 주는 지역을 결정한다.

NLSPATH
LC_MESSAGES의 처리를 위한 통보분류의 장소를 결정한다.

PATH system(cmd) 혹은 입출력파이프에 의하여 수행되는 지령들을 찾을 때 탐색경로를 결정한다.

모든 환경변수들은 awk의 변수 ENVIRON을 통하여 볼수 있다.

국제적인 코드모임보장

변수이름들이 ASCII기호들만 포함하여야 하며 정규식들은 유효한 기호들만 포함해야 한다는것을 제외하고는 단일바이트 및 다중바이트기호코드모임이 보장된다.

진단

awk는 매 기록에 대하여 199개 까지의 마당들($1, $2, …, $199)을 보장한다.

실례

72개 기호보다 긴 행들을 출력하려면 다음과 같이 하여야 한다.

```
length > 72
```

첫 두개의 마당을 반대순서로 출력하려면 다음과 같이 하여야 한다.

```
{ print $2, $1 }
```

반점이나 공백 그리고 태브에 의하여 입력마당들이 분리된다는것을 제외하고는 우와 같게 하려면 다음과 같이 하여야 한다.

```
BEGIN { FS = ", [ \t]*|[ \t]+ " }
      { print $2, $1 }
```

첫렬에 합과 평균값을 보충적으로 출력하려면 다음과 같이 하여야 한다.

```
       { s += $1 }"
END    { print "sum is", s, " average is", s/NR }
```

start/stop쌍들사이의 모든 행을 출력하려면 다음과 같이 하여야 한다.

```
/start/, /stop/
```

echo지령처럼 하려면 다음과 같이 하여야 한다(echo(1)을 참고).

```
BEGIN  {                                  #echo(1)과 류사하다.
     for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
```

```
printf "\n"
exit }
```

저자

awk는 AT&T, IBM, OSF, HP에 의하여 개발되였다.

관련항목

lex(1), sed(1)

A.V.Aho, B.W.Kernighan, P.J.Weinberger: The AWK Programming Language, Addision-Wesley, 1988

표준일치

awk: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

grep

grep - 주어 진 패턴을 탐색한다.

---

grep(1)

이름

grep, egrep, fgrep - 파일에서 패턴을 탐색한다.

형식

grep [-E|-F] [-c|-l|-q] [-insvx] pattern [file …]

grep [-E|-F] [-c|-l|-q] [-binsvx] -e pattern … [-e pattern] … [file …]

grep [-E|-F] [-c|-l|-q] [-insvx] [-f pattern_file] [file …]

절대형식:

egrep [-cefilnsv] [expression] [file …]

fgrep [-cefilnsvx] [strings] [file …]

해설

grep지령은 입력본문파일(기정값으로는 표준입력장치)에서 행별로 패턴을 탐색한다. 패턴이 들어 있는 행들은 표준출력장치에 보관된다. grep는 기본적인 정규식문법(regexp(5)를 참고), -E(egrep)는 **확장정규식**(Extended Regular Expression)을 보장한다. -F(fgrep)는 고속기호렬탐색알고리듬을 리용하여 고정된 기호렬을 탐색한다. -E와 -F들은 여러가지 기호패턴으로 새행들을 인식한다. 비여 있는 식이나 빈 기호렬들은 모든 행에 **정합**(Matching)된다.

egrep와 fgrep는 이전방안과의 호환성을 위한것이다.

선택항목들

-E 확장정규식들. 규정된 매 패턴은 한개이상의 ERE들의 렬이다. ERE들은 행바꾸기기호 혹은 -e로 주어 진 분리기호에 의하여 구별된다. ERE가 뒤에 행바꾸기기호가 없는 입력행의 내용과 일치될 때 매개 패턴은 그 입력행을 취한다. 동일한 기능을 egrep로 얻을수도 있다.

-F 고정된 기호렬들을 규정한다. 규정된 매개 패턴은 한개이상의 기호렬들의 렬이다. ERE들은 행바꾸기기호 혹은 -e로 주어 진 분리기호에 의하여 구별된다. 행이 그 렬안에 있는 임의의 기호렬들을 포함할 때 매개 패턴은 그 입력행을 취한다. 동일한 기능을 fgrep로 얻을수도 있다.

-b 매개 행은 앞에 그것의 블로크번호를 가진다. 블로크번호들은 파일에서 512개의 바이트크기로 구분된다.

-c 탐색된 행들의 개수만을 출력한다.

-e expression

한개의 식으로 된 인수와 같지만 기호 "-"로 시작되는 식인 경우에 효과적이다. 여러개의 -e로써 여러개의 패턴들을 규정할수 있다. 즉 입력행은 여러개 패턴들중의 어느 한개의 패턴과 일치되면 선택된다.

-f pattern_file

pattern_file로부터 정규식(grep와 grep -E) 혹은 기호렬목록(grep -F)을 취한다.

-i 비교프로쎄스에 대소기호차이를 무시한다.

-l 탐색된 행들을 가지는 파일이름들이 한번만 매행에 출력된다. 표준입력장치가 탐색되면 −의 경로이름이 출력된다.

-n 파일에서 매개 행은 앞에 1부터 시작하는 자기의 행번호를 가진다. 이 선택항목은 -c, -b, -l, -q가 있을 때 무시된다.

-q 탐색된 행에 무관계하게 표준출력장치에 아무것도 출력하지 않는다. 정합되는 행을 처음으로 발견하면 0상태를 가지고 탈퇴한다. 그리고 모든 선택항목들을 금지한다.

-s 파일이 존재하지 않거나 읽기불가능하면 오유통보를 내보낸다.

-v 정합되지 않은 모든 행들을 출력한다.

-x 전체 입력행들이 고정된 기호렬이나 정규식과 정확히 일치될

때에만 정합된것으로 본다.

출력이 생성되는 모든 경우에는 한개이상의 입력파일이 있을 때 파일이름이 출력된다. $, *, [, ^, |, (,), \과 같은 기호들을 식에서 리용할 때 전체 식을 웃반점들사이('')에 포함시키는것이 안전하다.

외부적영향

환경변수들

LANG은 LC_ALL과 LC_으로 시작하는 대응하는 환경변수들이 지역을 규정하지 않았을 때 지역분류를 위한 지역을 결정한다.

LANG이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG대신에 "C"가 기정값으로 리용된다(lang(5)를 참고).
LC_ALL은 LANG이나 LC_으로 시작하는 환경변수들의 설정으로 규정된 지역분류에 리용되는 임의의 값들을 무시하기 위한 지역을 설정한다.

LC_COLLATE는 정규식을 평가하는데 리용되는 기호코드모임을 결정한다.

LC_CTYPE는 단일/다중바이트기호로서의 본문에 대한 해석, 글자기호들의 분류, -i선택항목에서의 대소기호정보, 정규식에서 기호클라스로 탐색되는 기호들을 결정한다.

LC_MESSAGES는 통보가 현시되는 언어를 결정한다.

어떤 국제화변수가 틀린 설정을 포함하면 grep는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

귀환값

0　　한개이상의 탐색이 성공하였다.
1　　하나도 발견하지 못하였다.
2　　문법오유 혹은 접근불가능한 파일이다.

실례

Bourne쉘에서 2개의 파일로부터 4개의 기호렬들중의 임의의것을 포함하는 행들을 모두 찾으려면 다음과 같이 하여야 한다.

```
grep -F 'if
then
else
fi 'file1, file2
```

C쉘에서 우와 같은 작업을 하려면 다음과 같이 하여야 한다.

```
grep -F 'if then\else\ fi ' file1, file2
```

다음과 같은 4개의 주소가 있다.

```
Ken     112 Warring St. Apt. A
Judy    387 Bowditch Apt. 12
Ann     429 Sixth St.
```

다음의 지령은 Judy를 포함하는 행을 탐색해 낸다.

```
grep Judy address
```

Dec 혹은 Nov를 포함하는 행들을 탐색하려면 다음과 같이 하여야 한다.

```
grep -E '[Dd]ec|[Nn]ov' file
egrep -i 'dec|nov' file
```

현재 등록부안의 모든 파일들중에서 기호렬 xyz를 탐색하려면 다음과 같이 하여야 한다.

```
grep xyz *
```

현재 등록부의 부분나무안에 있는 모든 파일들에서 기호렬 xyz를 탐색하고 파일이름들이 체계인수목록의 제한조건을 넘지 않도록 하려면 다음과 같이 하여야 한다.

```
find .-type f -print |xargs grep xyz
```

앞의 실례에서는 기호렬 xyz가 나타나는 파일이름을 출력하지 않는다. 지령행의 grep지령부분에 다음과 같이 두번째 인수를 주면 파일이름들이 출력된다. 이때 첫번째 파일이름은 find에 의하여 생성되며 두번째 파일이름은 null파일이다.

```
find .-type f -print |xargs grep xyz /dev/null
```

경고

XPG4에서만은 -q가 있는 경우에 탈퇴상태가 0으로 된다. 그밖의 경우에는 표준적인 작용들이 적용된다.

관련항목

sed(1), sh(1), regcomp(3C), environ(5), lang(5), regexp(5),

표준일치

grep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
egrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
fgrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

sed

sed - 흐름식본문편집기.

---

sed(1)

이름

sed - 흐름식본문편집기

형식

sed [-n] script [file …]
sed [-n] [-e script] … [-f script_file] … [file …]

해설

sed는 본문파일(기정값으로는 표준입력장치)을 100개까지의 지령을 포함할수 있는 script에 따라 편집하여 표준출력장치에 복사한다. 완성된 입력행들만이 출력된다. 입력본문이 끝에서 행바꾸기기호로 끝나지 않으면 그것은 무시된다.

선택항목들

-f script_file
script_file로부터 스크립트를 취한다.

-e script script에 따르는 편집을 한다. 만일 -f가 없고 -e만 있으면 -e는 생략될수 있다.

-n 기정적인 출력장치를 금지한다.

sed는 모든 스크립트인수들을 주어 진 순서대로 처리한다. -e와 -f가 혼합되여 있으면 인쇄할수 없거나 부정확한 결과를 무시한다.

지령형식

스크립트는 편집지령들을 한 행에 하나씩 다음과 같은 형식으로 구성한다.

[address [，address]] function [arguments]

표준적인 연산에서 sed는 입력의 행들을 주기적으로(D지령뒤에 남아 있는것이 없다면) 패턴공간에 복사하고 그 패턴공간에 대하여 모든 지령들을 순차적으로 적용하며 마지막으로 패턴공간을 표준출력장치에 복사하고 패턴공간은 제거한다.

일부 지령들은 패턴공간의 일부 혹은 모두를 다음번 탐색에 리용하기 위하여 어떤 취해 진 공간을 리용한다.

지령주소

주소는 파일에서 입력행들을 계수하는 10진수이거나 입력의 마지막행을 가리키는 $ 또는 /정규식/형식의 문맥주소가 될수 있다.

- 문맥주소에서 \?정규식?은 /정규식/과 동등하다. 또한 문맥주소 \xabc\xdefx에서 두번째 x는 그자체이며 결국 정규식은 abcxdef이다.
- 탈퇴렬 \n은 패턴공간에 들어 온 행바꾸기기호와 정합된다.
- 점(.)은 패턴공간의 마지막 새행을 제외하고는 임의의 기호를 들여 맞춘다.
- 주소가 없는 지령행은 매개 패턴공간을 선택한다.
- 하나의 주소를 가지는 지령행은 그 주소에 들어 맞는 매개 패턴공간을 선택한다.
- 두개의 주소를 가지는 지령행은 첫번째 주소에 들어 맞는 패턴공간으로부터 두번째 주소에 들어 맞는 다음번 패턴공간까지의 배타적인 령역을 선택한다(만일 두번째 주소가 첫번째로 선택된 행번호보다 작거나 같다면 하나의 행만이 선택된다.). 결국 첫번째 주소를 가지고 과제가 반복된다.

sed는 기본정규식문장론을 보장한다(regexp(5)를 참고).
편집지령들은 부정함수 !에 의하여 선택되지 않은 패턴공간들에만 적용된다.

지령함수들

다음의 함수목록들에서 매개 함수들에 대하여 가능한 주소의 최대수는 ( )안에 지적되여 있다. 기타 함수성원들은 다음과 같이 해석된다.

text	한개이상의 행들이다. 여기서 마지막행은 행바꾸기기호를 숨기기 위하여 \으로 끝난다. \은 본문에서 s지령에 의한 기호렬교체의 \으로 해석되며 초기의 공백과 태브들을 방지하는데 리용된다.
rfile	지령행을 끝내야 하며 앞에는 정확히 하나의 공백이 있어야

한다.

wfile　　지령행을 끝내야 하며 앞에는 정확히 하나의 공백이 있어야 한다. 매개 wfile은 처리가 시작되기전에 창조된다. 최대로 10개까지의 서로 다른 인수들을 가질수 있다.

sed는 다음과 같은 함수들을 인식한다.

(1)a\
text　　추가. 다음 입력행을 읽기전에 출력에 본문을 놓는다.

(2)b label
　　label에 관계되는 :지령에로 갈래를 조성한다. label이 규정되지 않으면 스크립트의 끝에 간다.

(2)c\
text　　변경. 패턴공간을 제거한다. 0이거나 1인 주소 혹은 이 주소령역의 끝에서 본문을 표준출력장치에 넣는다. 그리고 다음순환을 시작한다.

(2)d　　패턴공간을 제거하고 다음의 순환을 계속한다.

(2)D　　첫번째 새행까지의 패턴공간의 초기토막을 제거하고 다음순환을 시작한다.

(2)g　　패턴공간의 내용을 취해 진 공간의 내용으로 교체한다.

(2)G　　취해 진 공간의 내용을 패턴공간에 추가한다.

(2)h　　취해 진 공간의 내용을 패턴공간의 내용으로 교체한다.

(2)H　　패턴공간의 내용을 취해 진 공간에 추가한다.

(1)i\
text　　삽입. 표준출력장치에 본문을 넣는다.

(2)l　　패턴공간을 표준출력장치에 렬거한다. 인쇄불가능한 기호들은 \뒤에 3개의 8진수로 표시한다.

(2)n　　기정의 출력장치가 금지되지 않았으면(지령행에 있는 선택항목 -n이 스크립트파일에 있는 #n지령에 의하여) 패턴공간을 표준출력장치에 복사한다. 패턴공간을 다음의 입력행과 교체한다.

(2)N　　다음의 입력행을 행바꾸기기호와 함께 패턴공간에 추가한다(현재 행번호는 변한다.).

(2)P　　인쇄. 패턴공간을 표준출력장치에 복사한다.

(2)p　　첫번째 새행까지의 패턴공간의 초기토막을 표준출력장치에로 복사한다.

(1)q　　탈퇴. 스크립트의 끝으로 간다. 새 순환을 시작하지 않는다.

(1)r rfile

rfile의 내용을 읽고 다음의 입력행을 읽기전에 그것을 출력장치에 넣는다.

(2)s/regular expression/replacement/flags

패턴공간에서 매개 regular expression을 replacement기호렬로 치환한다. 임의의 기호가 /대신에 쓰일수 있다. flags는 다음과 같다.

n　　n은 1부터2048(LINE_MAX)까지 가능하다. 패턴공간에서 n번째로 나타나는 regular expression만을 치환한다.

g　　대역적이다. 첫번째 발생을 제외한 모든 regular expression의 발생을 치환한다.

p　　교체작업이 진행되고 기정적인 출력장치가 금지되였을 때(지령행에 있는 -n나 스크립트파일에 있는 #n 지령에 의하여) 패턴공간을 인쇄한다.

w wfile　　쓰기작업을 규정한다. 교체작업이 진행되였다면 패턴공간을 wfile에 추가한다.

(2)t label

검사. 가장 최근의 입력행읽기나 t의 수행으로부터 어떤 치환이 진행되였다면 label과 관련된 :지령에로 갈래를 조성한다. label이 비여 있으면 스크립트의 끝으로 간다.

(2)w wfile

쓰기. 패턴공간을 wfile에 추가한다.

(2)x　　패턴의 내용을 취해 진 공간과 교체한다.

(2)y/string1/string2/

전환 .string1에 있는 기호들의 모든 발생을 string2의 대응하는 기호와 교체한다. 두 기호렬의 길이는 같아야 한다.

(2)! function

주소에 의하여 선택되지 않는 행에만 function을 적용한다.

(0): label

b와 t지령에 의한 갈래조성을 위한것으로써 아무것도 하지 않는다.

(1) = 　현재 행번호를 표준출력장치에 한개의 행으로 놓는다.

(2) 　{패턴공간이 선택되였을 때에만}까지의 아래의 명령문들을 수행한다.

문법형식은 다음과 같다.

```
{cmd1
 cmd2
 cmd3
  ⁞
 }
```

(0) 　빈 지령은 무시된다.

(0)# 　스크립트파일의 첫행에 있는 첫 기호가 #로 나타나면 전체 행은 설명문으로 고찰된다. #뒤에 있는 기호가 n이라면 표준출력장치는 금지된다. 그리고 #n뒤에 있는 행의 나머지부분은 무시된다. 스크립트파일은 적어도 하나의 비설명문행을 포함해야 한다.

외부적영향

환경변수들

LANG은 설정되지 않았거나 비여 있는 국제화변수들에 대한 기정값을 제공한다. LANG이 설정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL이 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모든 값을 무시한다.

LC_CTYPE는 본문의 단일/다중바이트기호들과 인쇄가능한 기호들의 분류, 식에서 기호모임식들에 대한 해석을 결정한다.

LC_MESSAGES는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용의 위치와 표준적인 출력장치에 씌여 지는 비형식적인 통보의 지역을 결정한다.

NLSPATH는 LC_MESSAGES의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임 보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

지령행이나 쉘스크립트로부터 파일안에서 abc를 단순히 xyz로 치환하려면 다음과 같이 하여야 한다.

```
sed 's/abc/xyz/' file1 > file1.out
```

쉘이나 환경변수 var1과 var2를 리용하여 우와 같은 탐색과 교체를 진행하려면 다음과 같이 하여야 한다.

```
sed "s/$var1/$var2/" file1 >file1.out
```

혹은

```
sed 's/'$var1'/'$var2'/' file1 >file1.out
```

한개의 행에서 다중치환을 실현하려면 다음과 같이 하여야 한다.

```
sed -e 's/abc/xyz/' -e 's/lmn/rst/' file1
```

혹은

```
sed -e 's/abc/xyz/' \
-e 's/lmn/rst/' \
file1 >file1.out
```

경고

sed는 지령스크립트의 개수를 100보다 작게 제한한다.
취해진 공간을 8192기호까지로 제한한다.
sed는 본문파일만을 처리한다.

저자

sed는 OSF, HP에 의하여 개발되였다.

관련항목

awk(1), ed(1), grep(1), environ(5), lang(5), regexp(5)
sed: A Non-Interactive Streaming Editor tutorial in the Text Processing Users Guide

표준일치

sed: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제7장. find 지령

find지령에 대한 개념

find지령을 사용하면 UNIX나무구조를 탐색하면서 파일들을 찾을수 있다. 사용자는 체계의 뿌리준위나 임의의 지점에서 탐색을 시작하여 전체 계층나무를 탐색할수 있다. 이렇게 탐색하면서 파일을 찾은 다음에는 그 파일들에 대한 작업을 수행할수 있다.

find지령의 일반형식은 다음과 같다.

find path operators

여기서 path는 탐색을 시작하는 등록부이고 operators는 사용자가 규정할수 있는 한개이상의 선택항목들이다.

find지령을 가장 많이 사용하는 목적은 파일들의 목록을 생성하는데 있다. 점(.)으로 규정된 현재 작업등록부안에 있는 파일들의 목록을 생성하고 그 파일들을 현시하자면 다음과 같이 할수 있다.

```
# cd /home
#
# ls -l
total    3
drwxr-xr-x   3  col     users      1024  Nov  8  14:09  col
drwxr-xr-x   6  root    root       1024  Nov  8  14:08  ftp
drwxr-xr-x   6  root    root       1024  Nov8  14:08  httpd
#
# find . -print
.
./httpd
./httpd/apache
./httpd/apache/doc
./httpd/apache/doc/manual.ps.qz
./httpd/cgi-bin
./httpd/cgi-bin/HelpIndex
./httpd/cgi-bin/HelpScreen
./httpd/html
./httpd/html/dt
./httpd/html/dt/dt.html
./httpd/html/dt/dt.html.idx
```

```
./httpd/html/dt/dt.index
./httpd/html/dt/expert.gif
./httpd/html/dt/hint.gif
./httpd/html/dt/index.gif
./httpd/html/dt/info2.gif
./httpd/html/dt/note.gif
./httpd/html/dt/sysadm.gif
./httpd/html/dt/up.gif
./httpd/html/dt/warning.gif
./httpd/icons
./ftp
./ftp/bin
./ftp/bin/gzip
./ftp/bin/ls
./ftp/bin/tar
./ftp/bin/zcat
./ftp/etc
./ftp/etc/group
./ftp/etc/passwd
./ftp/lib
./ftp/pub
./col
./col/.bashrc
./col/.cshrc
./col/.login
./col/.profile
./col/lg
./col/lg/lg_layouts
./col/lg/lg_layouts/User
./col/lg/lg3_prefs
./col/lg/lg3_soundPref
./col/lg/lg3_startup
```

우의 실례에서 find는 /home등록부로부터 수행되고 있다. /home등록부에는 3개의 홈등록부들이 있으며 find지령으로 매개 홈등록부들의 계층나무를 탐색하고 있다.

그런데 find지령을 리용할 때 파일이나 등록부에 대한 적절한 허락을 가지지 못하면 다음과 같은 통보를 받을수 있다.

```
find: /var/spool/cron: Permission denied
```

이런 통보는 사용자가 자기의 홈등록부밖에서 작업하는 경우에 find지령이나 여러가지 지령들을 사용할 때 때때로 받을수 있다.

find지령에서는 원칙적으로 탐색하는 파일에 대한 경로를 지정해야 한다. 다음의 실례에서는 탐색하려는 파일의 이름을 name식에서 규정하고 있다.

```
# find /home -name ftp
/home/ftp
```

이 실례에서는 ftp를 찾기 위하여 경로 /home을 탐색하고 있다.

규정된 종류의 파일찾기

find지령을 리용하면 등록부를 포함시키지 않으면서 파일들로만 되여 있는 목록을 생성할수 있다. 다음의 실례에서는 우의 실례에서와 류사하지만 파일들만의 목록을 생성하고 있다. 이 작업은 f형태를 찾도록 규정하면 된다.

```
# find /home -type f -print
/home/httpd/apache/doc/manual.ps.gz
/home/httpd/cgi-bin/HelpIndex
/home/httpd/cgi-bin/HelpScreen
/home/httpd/html/dt/dt.html
/home/httpd/html/dt/dt.html.idx
/home/httpd/html/dt/dt.index
/home/httpd/html/dt/expert.gif
/home/httpd/html/dt/hint.gif
/home/httpd/html/dt/index.gif
/home/httpd/html/dt/info2.gif
/home/httpd/html/dt/note.gif
/home/httpd/html/dt/sysadm.gif
/home/httpd/html/dt/up.gif
/home/httpd/html/dt/warning.gif
/home/ftp/bin/gzip
/home/ftp/bin/ls
/home/ftp/bin/tar
/home/ftp/etc/group
/home/ftp/etc/passwd
/home/col/.bashrc
/home/col/.cshrc
/home/col/.login
/home/col/.profile
```

/hoine/col/lg/lg_layoutB/User
/home/col/lg/lg3_prefs
/home/col/lg/lg3_soundPref
/home/col/lg/lg3_startup

find지령에서는 이 실례에서처럼 f를 파일, b를 규정된 파일의 블로크, l을 기호적련결 등으로 type를 규정하면 해당한 형태의 구체레들만 찾을수 있다.

빈파일과 빈등록부찾기

Linux와 같은 UNIX변종들에서는 -empty연산자를 가지고 지령 find에서 빈 파일과 빈 등록부를 찾도록 하고 있다. 아래의 실례에서 준 결과는 너무 길어서 일부만 출력한것이다.

find / -empty -print
/lost+found
/var/adm/LST/analyse
/var/spool/lpd
/var/spool/news
/var/spool/uucp
/var/spool/mqueue
/var/spool/atjobs/. SEQ
/var/spool/atspool
/var/spool/cron
/var/spool/fax/outgoing/locks
/var/spool/fax/incoming
/var/spool/voice/incoming
/var/spool/voice/messages
/var/spool/rwho
/var/spool/uucppublic
/var/lib/LST/log
/var/lib/LST/analyse
/var/lib/LST/disks
/var/lib/LST/catalog
/var/lib/LST/conflicts
/var/lib/LST/saved
/var/lib/LST/replaced
/var/lib/LST/deleted
/var/lib/games

/var/local
/var/lock/subsys/inet
/var/lock/subsys/ipx
/var/lock/subsys/syslog
/var/lock/subsys/amd
/var/lock/subsys/cron
/var/lock/subsys/atd
/var/lock/subsys/mta
/var/lock/subsys/rstatd
/var/lock/subsys/httpd
/var/log/httpd/apache/access_log
/var/log/xferlog
/var/log/uucp
/var/log/secure
/var/log/spooler
/var/named
/var/nis
/var/preserve
/var/run/xlaunch
/var/tmp
≀
/proc/net/snmp
/proc/net/raw
/proc/net/igmp
/proc/net/arp
/proc/net/unix
/proc/cpuinfo
/proc/pci
/proc/version
/proc/kmsg
/proc/meminfo
/Proc/uptime
/proc/loadavg
/proc/mdstat
/etc/modules/options
/etc/motd
/etc/exports
/etc/PPP/options
/tmp/LST

/tmp/. XF86Setup235/f5lcb27-315ae55/Serverout-2
/tmp/. XF86Setup246/13617839-llc7lc7l/Serverout-2
/tmp/. XF86Setup262/lf2ccOa9-aea3l4d/Serverout-2
/tmp/. XF86Setup288/17963ff7-21ca966f/Serverout-2
/tmp/fsslog
/mnt/floppy
/mnt/cdrom
/usr/doc/html/woven/LDP/install-guide-2. 2. 2. html/images. idx
/usr/lib/games
/usr/lib/groff/tmac/mm/locale
/usr/lib/groff/tmac/mm/se_locale
/usr/lib/kbd/keytables/patch_tables. orig
/usr/lib/perl5/i386-linux/5. 003/auto/GDBM_File/GDBM_File. bs
/usr/lib/perl5/i386-linux/5. 003/auto/DB_File/DB_File. bs
/usr/lib/perl5/i386-linux/5. 003/auto/Fcntl/Fcnti. bs
/usr/lib/perl5/i386-linux/5. 003/auto/FileHandle/FileHandle. bs
/usr/lib/perl5/i386-linux/5. 003/auto/NDBM_File/NDBM_File. bs
/usr/lib/perl5/i386-linux/5. 003/auto/POSIR/POSIX. bs
/usr/lib/perl5/i386-linux/5. 003/auto/SDBM_File/SDBM_File. bs
/usr/lib/perl5/i386-linux/5. 003/auto/Safe/Safe. bs
/usr/lib/perl5/i386-linux/5. 003/auto/Socket/Socket. bs
/usr/lib/perl5/i386-linux/5. 003/auto/Text/ParseWords
/usr/lib/perl5/site_perl/i386-linux
/usr/lib/linuxdoc-sgml/null. sty
/usr/etc
/usr/include/g++
/usr/include/netax25
/usr/include/netipx
/usr/include/readline
/usr/local/bin
/usr/local/doc
/usr/local/etc
/usr/local/games
/usr/local/info
/usr/local/lib
/usr/local/man/man1
/usr/local/man/man2
/usr/local/man/man3
/usr/local/man/man4

```
/usr/local/man/man5
/usr/local/man/man6
/usr/local/man/man7
/usr/local/man/man8
/usr/local/man/man9
/usr/local/man/mann
/usr/local/sbin
/usr/local/src
/usr/man/man9
/usr/man/mann
/usr/src/linux-2.0.29/include/linux/modules
/usr/src/linux-2.0.29/modules
/usr/src/redhat/BUILD
/usr/src/redhat/RPMS/i386
/usr/src/redhat/SOURCES
/usr/src/redhat/SPECS
/usr/src/redhat/SRPMS
/usr/X11R6/doc
/usr/X11R6/lib/X11/x11perfcomp
/usr/visix/fss/En_US.8859
/usr/visix/vls/En_US.8859
/usr/openwin/share/src/xview
/auto
/initrd
/home/httpd/icons
/home/ftp/lib
/home/ftp/pub
/opt/bin
/opt/man
/root/lg/lg3_hosts
/amd/nycaldi/auto/.vTRASH
#
# ll /auto
total 0
```

find지령에 대한 이 실레에서 출력된 파일 및 등록부들은 모두 비여 있다. 실레에서처럼 /auto에 의한 ll지령의 수행으로 이 사실을 알수 있다. 이러한 -empty는 모든 UNIX 변종들에서 항상 가능한것은 아니다.

이름, 크기, 이름과 크기에 의한 파일찾기

독자적인 찾기를 하여 그것들을 결합해 보자.

다음의 실례는 하나의 작은 탁상체계에서 수행된것이다. 사용자는 하나의 체계에서 뿐아니라 여러가지 조건에 맞는 전체 체계를 탐색할수도 있으며 보다 크고 복잡한 체계에서 작업하면서 전체 체계의 파일들을 찾을수도 있다. 또한 국부적인 기계에서만이 아니라 망준위에서 오랜 시간 여러 사용자들과 작업할수도 있다.

먼저 아래에서처럼 체계에서 .c로 끝나는 파일들을 모두 찾아 보자.

```
# find / -name *.c -print
/usr/X11R6/include/Xll/Xaw/Template.c
/usr/X11R6/include/Xaw3d/Template.c
/usr/X11R6/lib/X11/etc/et4000clock.c
/usr/doc/glibc-2.1.1/examples.threads/exl.c
/usr/doc/glibc-2.1.1/examples.threads/ex2.c
/usr/doc/glibc-2.1.1/examples.threads/ex3.c
/usr/doc/glibc-2.1.1/examples.threads/ex4.c
/usr/doc/glibc-2.1.1/examples.threads/ex5.c
/usr/doc/glibc-2.1.1/examples.threads/ex6.c
∫
/usr/doc/bind-8.2/notes/db_names.c
/usr/doc/1ibpng-1.0.3/example.c
/usr/doc/FAQ/htm1/c1one.c
/usr/doc/gnome-libs-devel/devel-docs/gnomeui/gnomeui-scan.c
#
```

우의 find지령의 수행결과를 보면 .c로 끝나는 파일들이 체계에 대단히 많다는것을 알수 있다.

우의 작업을 wc를 통하여 파이프로 련결시키면 아래에서 보는바와 같이 1,737개의 파일들이 .c로 끝나는 파일들이라는것을 알수 있다.

```
# find / -name *.c | wc
1737        1737    77044
#
```

이제 find지령으로 500,000개의 기호수가 넘는 크기의 파일전부를 찾아 보자.

```
# find / -size +500000c -print
/dev/core
/etc/X11/core
```

```
/var/lib/rpm/packages.rpm
/var/lib/rpm/fileindex.rpm
/var/lib/dosemu/hdimage.freedos
/var/lib/slocate/slocate.db
/var/lock/samba/SHARE_MEM_FILE
/proc/kcore
/home/ftp/lib/libc-2.1.1.so
/home/test/linuxi.xwd
/lib/libc-2.1.1.so
/lib/libdb-2.1.1.so
/lib/libm-2.1.1.so
/root/core
 ∫
/usr/share/gnome/help/users-guide/C/figs/full.gif
/usr/share/emacs/20.3/etc/DOC-20.3.1
/usr/share/emacs/20.3/leim/quail/ZIRANMA.el
/usr/share/emacs/20.3/leim/quail/ZIRANMA.elc
/usr/share/emacs/20.3/leim/skk/skkdic.el
/usr/share/emacs/20.3/leim/skk/skkdic.elc
/usr/share/emacs/20.3/lisp/loaddefs.el
/usr/share/sounds/card_shuffle.wav
/usr/share/sounds/startup2.wav
/usr/share/guavac/classes.zip
```

이 작업을 wc로 파이프로 련결시키면 215개의 파일이라는것을 알수 있다.

```
# find / -size +500000c -print | wc
215        215     6281
#
```

이제 우에서 리용한 두개의 지령을 결합하여 체계에서 .c로 끝나면서 동시에 500,000기호크기를 넘는 파일이 몇개인가를 알아 보자.

```
# find / -name *.c -size +500000c -print
/usr/src/drivers/scsi/advansys.c
#
```

1,737개의 파일이 .c로 끝나고 215개의 파일이 500,000개의 기호크기를 넘지만 이 두 조건을 만족시키는 파일은 오직 하나 advansys.c뿐이다. 우의 실례에서는 find지령앞에 and연산자가 암시적으로 작용하고 있다. 그러나 find에서 or연산자를 리용할수도 있지만 암시적으로 작용하지는 않는다.

만일 500,000기호수보다 크면서 .c와 .a로 끝나는 파일들을 찾으려고 한다면 어떻게 할것인가? 연산자 -o는 "or"를 의미하기때문에 .c와 .a로 끝나는 파일들을 지적하는데 리용할수 있다. 다음의 실례에서 이에 대하여 보여 주고 있다.

```
find / -size +500000c     -name *.c -o -name *.a \) -print
find / -size +500000c     -name *.c -o -name *.a \) -print | wc
```

```
# find / -size +500000c \( -name *.c -o -name *.a \ ) -print
/usr/X11R6/lib/libMagick.a
/usr/X11R6/lib/libX11.a
/usr/lib/bind/lib/libbind.a
/usr/lib/bind/lib/libbind_r.a
/usr/lib/libbfd.a
/usr/lib/libBLT.a
/usr/lib/libstdc++-2-libc6.1-1-2.9.0.a
/usr/lib/libc.a
 ⁝
/usr/lib/libinn.a
/usr/lib/libtiff.a
/usr/lib/liblinuxconf.a
/usr/lib/libform_g.a
/usr/lib/libncurses_g.a
/usr/lib/pythonl.5/config/libpythonl.5.a
/usr/lib/libpisock.a
/usr/lib/libcustoms.a
/usr/lib/libqt.a
/usr/lib/libst.a
/usr/src/drivers/scsi/advansys.c
#
```

찾으려는 두개의 파일확장자들은 괄호안에 놓인다. 그리고 괄호를 열거나 닫기전에 \이 반드시 놓인다. 왜냐하면 괄호는 쉘에서 특수한 의미가 있기때문에 find지령의것이라는것을 정확히 규정하여 쉘에서 리용되지 않도록 할 필요가 있다. 우에서 볼수 있는바와 같이 .c와 .a로 끝나면서 500,000기호수이상의 크기를 가지는 파일들은 많다.

우의 작업을 wc에 통과시켜 이 두가지 조건을 다 만족시키는 파일들이 정확히 몇개인가를 볼수 있다.

```
# find / -size +500000c \( -name *.c -o -name *.a \) -print | wc
    39      39      982
#
```

39개라고 결과가 나왔는데 앞선 실례에서 얻은 정보를 고려하면 .a로 끝나는 파일은 38개라는것을 알수 있다.

소유자, 종류, 허락에 의한 파일찾기

체계에서 특정한 사용자나 특정한 그룹이 소유하는 파일들을 찾아야 할 필요가 있을수 있다. "news"사용자가 소유하는 모든 **객체**(Object)들을 체계에서 찾자면 다음의 지령을 리용해야 한다.

```
# find / -user news -print
/etc/rc.d/rc.news
/etc/news
/etc/news/cleanfeed.conf
/etc/news/actsync.cfg
/etc/news/actsync.ign
/etc/news/control.ctl
/etc/news/cycbuff.conf
/etc/news/distrib.pats
/etc/news/expire.ctl
/etc/news/incoming.conf
/etc/news/inn.conf
/etc/news/innfeed.conf
/etc/news/innreport.conf
/etc/news/innwatch.ctl
/etc/news/moderators
/etc/news/motd.news
/etc/news/news2mail.cf
/etc/news/newsfeeds
/etc/news/nnrp.access
/etc/news/nnrpd.track
/etc/news/nntpsend.ctl
/etc/news/overview.ctl
/etc/news/overview.fmt
/etc/news/passwd.nntp
/etc/news/storage.conf
⋮
/usr/man/man8/tally.control.8
/usr/man/manB/tally.unwanted.8
/usr/man/man8/writelog.8
```

연산자 -user를 리용하면 사용자이름이나 사용자식별번호를 규정할수 있다.

다음의 실례에서는 "news"의 식별번호 "9"을 사용한 내용을 보여 주고 있다.

```
# find / -user 9 -print
/etc/rc.d/rc.news
/etc/news
/etc/news/cleanfeed.conf
/etc/news/actsync.cfg
/etc/news/actsync.ign
/etc/news/control.ctl
/etc/news/cycbuff.conf
/etc/news/distrib.pats
/etc/news/expire.ctl
/etc/news/incoming.conf
/etc/news/inn.conf
/etc/news/innfeed.conf
/etc/news/innreport.conf
/etc/news/innwatch.ctl
/etc/news/moderators
/etc/news/motd.news
/etc/news/news2mail.cf
/etc/news/newsfeeds
/etc/news/nnrp.access
/etc/news/nnrpd.track
/etc/news/nntpsend.ctl
/etc/news/overview.ctl
/etc/news/overview.fmt
/etc/news/passwd.nntp
/etc/news/storage.conf
 ⌠
/usr/man/man8/tally.control.8
/usr/man/man8/tally.unwanted.8
/usr/man/man8/writelog.8
```

이 find지령은 앞의 실례와 꼭 같은 결과를 생성하였다.

여러가지 파일종류 즉 f(파일), b(블로크), l(기호적련결) 등으로 type식을 꾸미면 파일종류에 따라 파일들을 탐색할수 있다. type -d는 "news"에 속하는 등록부들만 찾아낸다.

```
# find / -user news -type d -perm 775 -print
```

```
/etc/news
/var/lib/news
/var/log/news
/var/log/news/OLD
/var/spool/news
/var/spool/news/archive
/var/spool/news/articles
/var/spool/news/incoming
/var/spool/news/incoming/bad
/var/spool/news/innfeed
/var/spool/news/outgoing
/var/spool/news/overview
/var/spool/news/uniover
/var/spool/slrnpull
/usr/bin/auth
/usr/bin/control
/usr/bin/filter
/usr/bin/rnews. libexec
```

이 실례에서도 and가 암시적으로 작용하고 있다. 즉 find는 소유자가 "news"이고 파일종류가 등록부로 되는 대상들만을 현시하였다.

이제 여기에 또 하나의 조건 즉 허락조건까지 추가하여 and가 암시적으로 작용하게 할수 있다.

다음의 실례에서는 775라는 허락을 가진 등록부들중에서 "news"에 속하는 객체들만을 찾아 내고 있다.

```
# find / -user news -type d -perm 775 -print
/etc/news
/var/lib/news
/var/log/news
/var/log/news/OLD
/var/spool/news
/var/spool/news/archive
/var/spool/news/articles
/var/spool/news/incoming
/var/spool/news/incoming/bad
/var/spool/news/innfeed
/var/spool/news/outgoing
/var/spool/news/overview
```

/var/spool/news/uniover
/var/spool/slrnpull
/usr/bin/auth
/usr/bin/control
/usr/bin/filter
/usr/bin/rnews. libexec

이 실례에서 보는바와 같이 "news"에 속하는 소유자는 두명이며 이 그룹의 성원들은 읽기-쓰기-실행허락을 가지고 나머지성원들은 읽기-실행접근을 가진다는것을 알수 있다. 이것은 체계관리자들이 수행하는 보편적인 find연산의 한가지 형태이다. 즉 특정의 사용자들이 특정의 허락을 가지는 파일과 등록부를 찾기 위한 연산이다.

낡은 파일의 찾기와 찾은 파일에서의 작업

체계와 자기의 홈등록부들에는 오래동안 접근을 하지 않은 낡은 파일들이 존재할수 있다. 홈등록부에서 200일동안 접근하지 않은 파일들을 찾자면 다음과 같이 한다.

 # find . -atime +200 -print
. /hp/. Xdefaults
. /hp/. bash_logout
. /hp/. bash_profile
. /hp/. bashrc
. /hp/. kde/share/apps/kfm/desktop
. /hp/. kde/share/config/desktoporc
. /hp/. kde/share/config/kcmdisplayrc
. /hp/. kde/share/config/kfmrc
. /hp/. kde/share/config/kpanelrc
. /hp/. kde/share/config/kvtrc
. /hp/. kderc
. /hp/Desktop/. directory
. /hp/Desktop/Printer. kdelnk
. /hp/Desktop/Templates/Device. kdelnk
. /hp/Desktop/Templates/Ftpurl. kdelnk
. /hp/Deektop/Templates/MimeType. kdelnk
. /hp/Desktop/Templates/Program. kdelnk
. /hp/Desktop/Templates/URL. kdelnk
. /hp/Desktop/Templates/WWWUrl. kdelnk
. /hp/Desktop/cdrom. kdelnk
. /hp/Desktop/floppy. kdelnk

```
./hp/.screenrc
./hp/.bash_history
./test/.Xdefaults
./test/.bash_logout
./test/.bash_profile
./test/.bashrc
./test/.kde/share/apps/kfm/desktop
./test/.kde/share/config/desktoporc
./test/.kde/share/config/kcmdisplayrc
./test/.kde/share/config/kfmrc
./test/.kde/share/config/kpanelrc
./test/.kde/share/config/kvtrc
./test/.kderc
./test/Desktop/.directory
./test/Desktop/Printer.kdelnk
./test/Desktop/Templates/Device.kdelnk
./test/Desktop/Templates/Ftpurl.kdelnk
./test/Desktop/Templates/MimeType.kdelnk
./test/Desktop/Templates/Program.kdelnk
./test/Desktop/Templates/URL.kdelnk
./test/Desktop/Templates/WWWUrl.kdelnk
./test/Desktop/cdrom.kdelnk
./test/Desktop/floppy.kdelnk
./test/.screenrc
./test/linuxi.xwd
./test/.bash_history
#
```

우에서 보는것처럼 결과는 많은 파일들을 현시하고 있다. 여기에 지령 ls -l을 추가하여 파일들에 대한 보충적인 정보를 얻자면 다음과 같이 하여야 한다.

 # find . -atime +200 -exec ls -l { } \;

```
-rw-r--r--   1 hp   hp   1422   Aug 28 ./hp/.Xdefaults
-rw-r--r--   1 hp   hp   24     Aug 28 ./hp/.bash logout
-rw-r--r--   1 hp   hp   230    Aug 28 ./hp/.bashDprofile
-rw-r--r--   1 hp   hp   124    Aug 28 ./hp/.bashrc
-rw-r--r--   1 hp   hp   260    Aug 28 ./hp/.kde/share/apps/kfm/desktop
-rw-r--r--   1 hp   hp   234    Aug 28 ./hp/.kde/share/config/desktoporc
```

```
-rw-r--r--   1 hp    hp    49    Aug 28 ./hp/.kde/share/config/kcmdisplayrc
-rw-r--r--   1 hp    hp    95    Aug 28 ./hp/.kde/share/config/kfmrc
-rw-r--r--   1 hp    hp    456   Aug 28 ./hp/.kde/share/config/kpanelrc
-rw-r--r--   1 hp    hp    255   Aug 28 ./hp/.kde/share/config/kvtrc
-rw-r--r--   1 hp    hp    966   Aug 28 ./hp/.kderc
-rw-r--r--   1 hp    hp    85    Aug 28 ./hp/Desktop/.directory
-rw-r--r--   1 hp    hp    222   Aug 28 ./hp/Desktop/Printer.kdelnk
-rw-r--r--   1 hp    hp    607   Aug 28 ./hp/Desktop/Templates/Device.kdelnk
-rw-r--r--   1 hp    hp    296   Aug 28 ./hp/Desktop/Templates/Ftpurl.kdelnk
-rw-r--r--   1 hp    hp    324   Aug 28 ./hp/Desktop/Templates/mimeType.kdelnk
-rw-r--r--   1 hp    hp    170   Aug 28 ./hp/Desktop/Templates/Program.kdelnk
-rw-r--r--   1 hp    hp    411   Aug 28 ./hp/Desktop/Templates/URL.kdelnk
-rw-r--r--   1 hp    hp    458   Aug 28 ./hp/Desktop/Templates/WWWUrl.kdelnk
-rw-r--r--   1 hp    hp    376   Aug 28 ./hp/Desktop/cdrom.kdelnk
-r-r--r--    1 hp    hp    378   Aug 28 ./hp/Desktop/floppy.kdelnk
-rw-rw-r--   1 hp    hp    3505  Aug 28 ./hp/.screenrc
-rw-------   1 hp    hp    34    Aug 28 ./hp/.bash_history
-rw-r--r--   1 test  test  1422  Aug 28 ./test/.Xdefaults
-rw-r--r--   1 test  test  24    Aug 28 ./test/.bash_logout
-rw-r--r--   1 test  test  230   Aug 28 ./test/.bash_profile
-rw-r--r--   1 test  test  124   Aug 28 ./test/.bashrc
-rw-r--r--   1 test  test  260   Aug 28 ./test/.kde/share/apps/kfm/desktop
-rw-r--r--   1 test  test  234   Aug 28 ./test/.kde/share/config/desktoporc
-rw-r--r--   1 test  test  49    Aug 28 ./test/.kde/share/config/kcmdisplayrc
-rw-r--r--   1 test  test  95    Aug 28 ./test/.kde/share/config/kfmrc
-rw-r--r--   1 test  test  456   Aug 28 ./test/.kde/share/config/kpanelrc
-rw-r--r--   1 test  test  255   Aug 28 ./test/.kde/share/config/kvtrc
-rw-r--r--   1 test  test  966   Aug 28 ./test/.kderc
-rw-r--r--   1 test  test  85    Aug 28 ./test/Desktop/.directory
-rw-r--r--   1 test  test  222   Aug 28 ./teBt/Desktop/Printer.kdelnk
-rw-r--r--   1 test  test  607   Aug 28 ./test/Desktop/Templates/Device.kdelnk
-rw-r--r--   1 test  test  296   Aug 28 ./test/Desktop/Templates/Ftpurl.kdelnk
-rw-r--r--   1 test  test  324   Aug 28 ./test/Desktop/Templates/mimeType.kdelnk
-rw-r--r--   1 test  test  170   Aug 28 ./test/Desktop/Templates/Program.kdelnk
-rw-r--r--   1 test  test  411   Aug 28 ./test/Desktop/Templates/URL.kdelnk
-rw-r--r--   1 test  test  458   Aug 28 ./test/Desktop/Templates/WWWUrl.kdelnk
-rw-r--r--   1 test  test  376   Aug 28 ./test/Desktop/cdrom.kdelnk
-rw-r--r--   1 test  test  378   Aug 28 ./test/Desktop/floppy.kdelnk
-rw-rw-r--   1 test  test  3505  Aug 28 ./test/.screenrc
```

```
-rwxrwxrwx 1  root   root      614955  Aug  28  ./test/linuxi.xwd
-rw-------    1  test   test      13          Aug  28  ./test/.bash-history
#
```

이 지령은 앞의 실례와 동일한 출력을 생성하는데 find의 출력을 { }가 있는 위치에 놓고 있다. ls -l은 find에 의해 생성된 파일이름을 리용하며 그것은 인수로서 { }안에 포함된다. 지령의 뒤에 ;이 있기때문에 \을 리용하여 쉘이 아니라 find지령에서 ;을 인식하도록 하고 있다.

find 개요

아래에서는 find에서 공통적으로 쓰이는 연산자들에 대한 개요를 주고 있다. UNIX 변종에 따라 -print가 필요하거나 필요하지 않을수 있다.

find – 파일들을 탐색한다.

공통적으로 쓰이는 연산자들과 선택항목들:

-atime n
: n일전에 접근한 파일들을 찾는다. +n은 n일보다 더 오래전에, -n은 m(<n)일전에 접근한 파일들을 찾는다.

-ctime n
: inode가 n일전에 수정된 파일들을 탐색한다. +n과 -n의 의미는 -atime n에서와 같다.

-exec command
: command로 규정한 지령들을 수행한다.

-group name
: name그룹안에 속하는 파일들을 찾는다. name은 그룹 ID번호일수도 있다.

-mount
: 다른 파일체계의 등록부들은 설치하지 않는다(모든 UNIX변종에서 가능한것이 아니다.).

-mtime n
: n일전에 수정된 파일들을 찾는다. +n은 n일보다 더 오래전에 수정한 파일들을 찾고 -n은 n일보다 작게 즉 m(<n)일전에 접근한 파일들을 찾는다.

-newer file

파일은 file보다 최근에 수정되였다.

-name pattern
pattern으로 주어 지는 파일이름을 찾는다.

-ok command
지령을 수행하기전에 수행이 가능한가를 검사한다.

-perm mode
규정된 접근방식 mode를 가지는 파일들을 찾는다. mode는 8진수로 주어야 한다.

-print
현재 파일이름을 표준출력장치에 출력한다.

-type t
d(등록부) 및 f(파일) 등과 같은 t종류의 파일들을 찾는다.

-size n
n블로크크기의 파일들을 찾는다. 한개의 블로크는 보통 512byte이다. +n은 n개보다 많은 블로크수, nc는 n개 기호수, +nc는 n보다 많은 기호수를 가진 파일들을 찾는다.

-user uname
uname이 소유하는 파일을 찾는다.

다중조건:
-a는 and연산자
-o는 or연산자
!는 연산자가 정합하지 못한다는것을 규정한다.
\(expression)\은 다른것이 규정되기전에 식 expression을 평가한다는것을 규정한다.

지령소개

아래에서는 이 장에서 리용한 지령을 보여 주고 있다. UNIX변종들에서 지령은 약간 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

find

find - 체계의 계층구조를 재귀적으로 탐색하면서 파일들을 찾는다.

find(1)

이름

find - 파일들을 찾는다.

형식

find pathname_list [expression]

해설

find지령은 주어 진 론리식에 들어 맞는 파일들을 탐색하면서 pathname_list에 있는 매 경로이름에 대한 등록부계층나무를 재귀적으로 감소시킨다.

아래에서 나타나는 n은 10진옹근수를 표시하며 +n은 n보다 크다는것을, -n은 n보다 작다는것을 표시한다.

선택항목들

-depth 위치에 무관계하게 작용하는 위치독립항으로서 등록부계층을 하나만큼 감소시켜 작업하게 한다. 즉 등록부안의 모든 구체례에 대한 작업이 그 등록부자체보다 먼저 진행되게 한다. 이것은 find가 cpio(1)을 리용하여 쓰기허락이 없는 등록부의 파일들을 전송하려고 할 때 효과적이다. 또한 이것은 cpio(1)을 리용하여 등록부의 수정날자를 보관하려고 할 때 효과적이다. 이것은 항상 정확한것이다.

-follow 위치독립항으로서 find가 기호적련결들을 찾지 않도록 한다. 이것은 항상 정확한것이다.

-fsonly FStype

위치독립항으로서 파일체계가 FStype로 규정된 종류가 아니면 탐색을 중지하도록 한다. FStype는 CDFS, HFS, NFS파일체계에 각각 대응하는 cdfs, hfs, nfs들중의 하나를 취한다.

계층나무의 매 정점은 자기 선조등록부의 FStype를 계승한다. 실례로 -fsonly hfs가 규정되였을 때 HFS파일체계에 있는 NFS의 정점을 발견하면 그 정점아래의 구체례들은 탐색하지 않는다. 결국 -fsonly nfs가 규정되였을 때 NFS파일체계의 아래에 있는 임의의 HFS파일체계들은 탐색되지 않는다는것을 주의해야 한다. 이것은 항상 정확한것이다.

-xdev 위치독립항으로서 pathname_list에서 취해 진 시작정점에 대하여 그 아래에 존재하는 임의의 파일체계정점들을 무시하도록 한다. 이것은 항상 정확한것이다.

-mountstop
-xdev와 동일하다. 이것은 이전 방안과의 호환성을 위하여 제공된다.

-name file
file이 현재파일이름의 마지막성분과 정합될 때 정확한것이다(정합조작은 regexp(5)를 참고.).

-path file
기본이름이 아니라 완전경로가 리용된다는것을 제외하고는 -name과 동일하다(경로이름은 -print에 의해 출력될수 있다.). 여기서 /은 특수한 의미로 해석되지 않는다. 실례를 들어 */. profile은 . /home/fred/. profile을 들어 맞춘다.

-perm [-]mode
mode는 파일의 방식비트들을 표시한다. 이 인수는 chmod(1)에서의 방식인수와 형식이 같다. 그러나 첫 기호는 −연산자가 되지 말아야 한다. 기호적형식을 리용할 때 출발모형은 모든 파일방식비트들이 청소되였다고 가정한다.

−가 없으면 파일허락비트들이 정확히 mode의 값과 일치될 때에만 성공한다. 기호적속성들인 s(사용자ID 설정, 그룹ID 설정)와 t(보호비트)에 관련되는 비트들은 -가 없을 때 무시된다.

mode앞에 -가 있으면 mode에 설정되는 비트들모두가 파일허락비트들에 설정될 때 성공한다. 이 경우에 기호적속성들인 s와 t에 관련되는 비트들은 의미가 있다.

-fstype FStype
파일이 속하는 파일체계가 FStype일 때 성공한다. 여기서 FStype는 CDFS, HFS, NFS파일체계에 각각 대응하는 cdfs, hfs, nfs들중의 하나를 취한다.

-type c 파일의 종류가 c일 때 성공한다. c는 다음과 같은것들중의 하나이다.

f	표준파일
d	등록부
b	블로크파일
c	기호파일
p	FIFO(파이프)
l	기호적련결

s　　소케트
n　　망파일
M　　나무의 정점

-links n　파일이 n개의 련결을 가지면 성공한다.

-user uname
파일이 사용자 uname에 속하는 파일이라면 성공한다. uname이 수값이고 /etc/passwd파일안에 있는 가입이름이 아니면 사용자ID를 취한다. uname에는 앞에 + 혹은 −가 붙을수도 있다.

-group gname
파일이 그룹 gname에 속하는 파일이라면 성공한다. gname이 수값이면서 /etc/passwd파일안에 있는 가입이름이 아니면 그룹ID를 취한다. gname에는 앞에 + 혹은 −가 붙을수도 있다.

-nouser　파일이 암호자료기지에 없는 사용자ID에 속하는 파일이라면 성공한다(passwd(4)를 참고).

-nogroup　파일이 그룹자료기지에 없는 그룹ID에 속하는 파일이라면 성공한다(group(4)를 참고).

-size n[c]
파일이 n개 블로크의 길이를 가지면 성공한다. c가 규정되면 바이트크기이다.

-atime n　파일이 n일동안에 접근한것이면 성공한다. pathname_list안에 있는 등록부들의 접근시간은 발견시간 그자체에 의하여 변경된다.

-mtime n
파일이 n일동안에 수정된것이면 성공한다.

-ctime n
파일의 inode가 n일동안에 변경된것이면 성공한다.

-newer file
현재 파일이 주어 진 file보다 최근에 수정된것이면 성공한다.

-nwer [tv1 [tv2]] file
현재 파일의 지적된 시간(tv1)이 file의 지적된 시간(tv2)보다 새로운것이라면 성공한다. tv1와 tv2은 다음의 기호들로부터 선택된다.

a　　파일이 마지막으로 접근된 시간
c　　파일의 inode가 마지막으로 수정된 시간
m　　파일이 마지막으로 수정된 시간

만일 tv2가 생략되면 기정값으로는 m을 취한다. -newer는 -newermm과

동등하다.

실례:

-newera file
-newermc file

-inum n 파일의 계렬번호(inode번호)가 n이면 성공한다. 파일의 계렬번호들은 주어 진 파일체계에서 유일하다. 파일계렬번호들을 찾는것은 참조된 파일들이 단일한 파일체계만 찾도록 제한하지 않는 조건에서는 같다는 것을 담보하지 않는다.

-linkedto path
파일이 path(즉 path에 련결된)에 의하여 규정된 파일과 물리적으로 같을 때 성공한다. 이것은 -inum과 류사하지만 파일이 path에 하드련결되였는가를 다중파일체계에서 탐색할 때에도 정확히 검사한다.

-print 현재경로이름이 출력되도록 한다. 이것은 항상 정확한것이다.

-exec cmd
만일 수행된 cmd가 0값을 가지고 탈퇴하였다면 성공한다. cmd의 끝은 ;이여야 한다(;은 쉘에서 특수하며 탈퇴되여야 한다.). 임의의 지령인수 { }은 현재 경로이름에 의하여 교체된다.

-ok cmd
생성된 지령행이 처음에 ?를 인쇄하고 사용자가 y로 대답했을 때만 수행된다는것을 제외하고는 -exec와 같다.

-cpio device
현재 파일을 cpio(4)형식(517byte기록들)으로 device에 출력한다. -cpio의 리용은 -depth를 암시한다. 이것은 항상 정확한것이다.

-ncpio -c를 cpio에 추가한다는것을 제외하고 -cpio와 동일하다. -ncpio의 리용은 -depth를 암시한다. 이것은 항상 정확한것이다.

-prune 현재의 구체례가 등록부이면 find는 그 등록부를 뛰여 넘는다. 이것은 일정한 등록부들을 피하거나 cpio -p로서 일부 재귀적인 등록부들을 피하면서 작업할 때 효과적이다.

-only 이것은 -prune에 대한 정값론리의 방안이다. -only가 매 등록부에 대하여 성공하지 못하면 그 등록부의 뒤에서 -prune가 수행된다. 실례로 다음의 3개 지령은 동등하다.

find .-fsonly hfs -print
find .-print -fstype hfs -only
find .-print -fstype hfs -prune

이것은 항상 정확한것이다.

(expression)

expression이 정확한것이면 성공한다. 괄호는 쉘에서 특수한 의미를 가지는 \와 \에 의하여 탈퇴되여야 한다.

우에서 고찰한 선택항목들은 다음의 연산자들에 의하여 결합될수 있다.

! expression

론리적인 NOT연산자. expression이 부정확할 때 정확한것이다.

expression [-a] expression

론리적인 AND연산자. 두개의 expression이 다 정확한것일 때 정확한것이다.

expression -o expression

론리적인 OR연산자. 두개의 expression중에서 하나가 정확한것일 때 정확한것이다.

만일 expression이 생략되거나 -print, -ok, -exec, -cpio, -ncpio가 아니면 -print로 가정한다.

접근조종목록

접근조종목록선택항목들은 사용자가 접근조종목록구체례들을 찾을수 있게 한다. 이것은 파일의 접근조종목록이 접근조종목록패턴을 들어 맞추거나 접근조종목록의 구체례를 선택적으로 포함한다면 성공이다(acl(5)를 참고). 다음과 같은 3개의 형태가 있다.

-acl aclpatt

aclpatt패턴에 의하여 규정된 모든 패턴구체례들을 포함하는 접근조종목록을 가지는 모든 파일들을 들어 맞춘다.

-acl = aclpatt

aclpatt패턴에 의하여 규정된 모든 패턴구체례들을 포함하는 접근조종목록을 가지는 하나의 파일만을 들어 맞춘다. 여기서 접근조종목록안에 있는 매개 구체례는 aclpatt패턴에서 규정된 적어도 하나의 패턴구체례에 정합된다.

-acl opt

선택적인 접근조종목록구체례들을 포함하는 모든 파일들을 들어 맞춘다.

aclpatt기호들은 하나의 연산자 혹은 짧은 형식의 패턴으로 주어질수 있다(acl(5)를 참고).

표준적으로 -acl은 aclpatt에 있는 모든 접근조종목록패턴들을 포함하는 접근조종목록을 가지는 파일에 대하여 성공한다. 파일의 접근조종목록은 정합되지 않는 구체례들도 포함할수 있다.

aclpatt는 patt가 =로 시작하면 나머지기호렬은 파일의 접근조종목록에 있는 모든 구체례와 정합되여야 한다.

aclpatt기호렬은 접근조종목록 혹은 접근조종목록패턴이 될수 있다. 만일 그것이 접근조종목록이라면 aclpatt는 반드시 적어도 3개의 기본구체례들인 ((user.%, mode), (%.group, mode), (%.%, mode))를 포함하여야 한다.

특수한 경우로서 aclpatt가 opt라면 선택항목은 접근조종목록구체례들을 가진 파일에 대하여 성공한다.

외부적영향

환경변수들

LANG은 설정되지 않았거나 비여 있는 국제화변수들에 대한 기정값을 제공한다.

LANG이 설정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고).

LC_ALL이 비지 않은 기호렬값으로 설정되여 있다면 다른 국제화변수들의 모든 값을 무시한다.

어떤 국제화변수가 틀린 설정을 포함하면 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_CTYPE는 본문의 단일/다중바이트기호들, 인쇄할수 있는 기호들의 분류, 탐색패턴에 있는 기호클라스식들에 의하여 정합되는 기호들에 대한 해석을 결정한다.

LC_MESSAGES는 표준적인 오유장치에 씌여 지는 진단통보의 형식화된 내용의 위치와 표준적인 출력장치에 씌여 지는 비형식적인 통보에 영향을 주는 지역을 결정한다.

NLSPATH는 LC_MESSAGES의 처리를 위한 통보분류의 장소를 결정한다.

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

두개의 등록부들인 /example과 /new/example에서 기호렬 Where are you를 포함하는 파일들을 탐색하여 그 파일들의 이름을 출력하려면 다음과 같이 하여야 한다.

```
find /example /new/example -exec grep -l 'Where are you' { } \;
```

a.out나 *.o라는 이름을 가진 파일들중에서 한주일동안 접근을 하지 않은 파일들을 제거하려면 다음과 같이 하여야 한다.

find / \(-name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
여기서 탈퇴괄호들앞에 있는 공백은 반드시 필요한것이다.

콤퓨터에 있는 모든 파일들의 이름을 출력하고 nfs정점들을 출력할 때에는 nfs등록부들을 피하도록 하자면 다음과 같이 하여야 한다.

find / -fsonly hfs -print

전체 파일체계를 설치된 /Disk정점에 복사하고 재귀적인 복사문제를 피하려면 다음의 동등한 두개의 명령문들을 사용할수 있다.

cd /; find .! -path ./Disk -only -print | cpio -pdxm /Disk
cd /; find .-path ./Disk -prune -o -print | cpio -pdxm /Disk

뿌리디스크를 설치된 /Disk정점에 복사하되 /.아래의 모든 파일체계들을 뛰여 넘도록 해보자.-xdev는 /이 정점이라 해도 뛰여 넘지 못하게 한다. 왜냐하면 /은 시작정점이고 -xdev는 시작정점아래의 구체례들만 처리하기때문이다.

cd / ; find .-xdev -print | cpio -pdm /Disk

등록부부분나무에 있는 모든 표준파일들에서 허락들을 444로 변경하고 모든 등록부들에 대한 허락을 555로 변경하려면 다음과 같이 하여야 한다.

find <pathname> -type f -print | xargs chmod 444
find <pathname> -type d -print | xargs chmod 555
find로부터의 결과는 -exec대신에 xargs(1)에로 파이프련결되였다. 왜냐하면 많은 개수의 파일이나 등록부가 하나의 지령으로 처리될 때 -exec는 매개 파일이나 등록부에 대하여 서로 다른 처리를 하며 한편 xargs는 파일이름과 등록부이름들을 하나의 chmod지령의 여러 인수들에 수집해 넣기때문이다.

접근조종목록실례

사용자 kar1이 소유하지 않는 파일전부를 찾아 보자.이 파일들은 kar1과 관련되는 적어도 하나의 구체례가 있는 접근조종목록을 가지며 읽기비트는 on이고 쓰기비트는 off인 bin그룹안의 아무런 성원도 이 구체례를 리용할수 없다.

find / ! user kar1 -acl 'kar1.*, %.bin+r-w' -print

임의의 접근조종목록구체례에서 설정된 읽기비트를 가지는 파일들을 모두 찾자면 다음과 같이 하여야 한다.

find / -acl '*.*+r' -print

모든 접근조종목록구체례에서 설정되지 않은 쓰기비트와 설정된 실행비트를 가지는

파일들을 모두 찾자면 다음과 같이 하여야 한다.

```
find / -acl '=*.*-w+x' -print
```

선택적인 접근조종목록구체레를 가지는 파일들을 모두 찾자면 다음과 같이 하여야 한다.

```
find / -acl opt -print
```

종속성

NFS
-acl는 NFS파일들에서 항상 부정확한것이다.

경고

내부연산과 관련하여 cpio는 2Gbyte이상으로 큰 파일들이거나 60,000(60Kbyte)이상의 사용자/그룹ID를 가지는 파일들은 제공하지 않는다. 60Kbyte보다 큰 사용자/그룹ID를 가진 파일들은 현재 과제의 사용자/그룹ID의 아래에서 관리되고 재기억된다.

저자

find는 AT&T와 HP에 의하여 개발되였다.

파일

/etc/group	그룹이름/etc/mnttab 설치된 정점들
/etc/passwd	사용자이름

관련항목

chacl(1), chmod(1), cpio(1), sh(1), test(1), xargs(1), mknod(2), stat(2), cpio(4), fs(4), group(4), passwd(4), acl(5), environ(5), lang(5), regexp(5)

표준일치

find: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제8장. vi 편집기

vi 편집기

대부분의 UNIX사용자들은 도형사용자대면부(GUI)를 통하여 자기의 UNIX체계와 접근한다. 이러한 GUI가 공동탁상환경(CDE)으로 된다. CDE는 발전된 창문환경을 제공하는 X Windows체계와 Motif에 기초하고 있다.

많은 UNIX GUI들은 도형식편집기를 제공한다. **도형식편집기**(Graphical Editor) vi는 가장 보편적인 UNIX편집기로 되고 있지만 그것이 GUI의 전부인것은 아니다. UNIX GUI의 도형식편집기들이 표준적으로 제공되는것도 많고 PC용창문환경의 편집기들도 많이 존재하지만 이 장에서 vi를 취급하는 리유는 다음과 같다.

- UNIX체계를 사용하는 사람들모두가 **도형현시장치**(Graphics Display)에 대한 접근을 하지는 않으므로 vi를 알고 리용할 필요가 있다. 왜냐하면 vi가 UNIX토대의 강력한 편집기이기때문이다(현재는 강력한 도형사용자대면부의 제공으로 하여 새로운 UNIX사용자들은 vi를 사용하지 않는다.).
- vi는 전문적으로 사용하여 온 UNIX편집기이다. UNIX사용자들의 일부는 vi를 리용해 보지 못하였지만 대부분은 vi에 대한 경험을 가지고 있다.

ed라는 행편집기도 UNIX체계들에서 많이 리용되였지만 vi가 화면편집기인것으로 하여 행편집기 ed는 드물게 리용되고 있다(ed의 확장방안은 ex이다.).

이 장에서는 vi를 리용하는 기본수법과 몇가지 발전된 기능들에 대하여 서술한다.

이 장에서 제공되는 표들은 vi에서 공통적으로 사용되는 지령들을 개괄하고 있다. 표 8-1에서는 이러한 표들에 대한 목록을 보여 주고 있다.

표 8-1　　메타기호들과 그 리용프로그람들

표번호	vi함수
Expr	정규식
Introduction	vi의 방식과 표시법
1	vi대화조종의 시동
2	vi에서 유표조종지령들
3	vi에서 본문의 추가
4	vi에서 본문의 제거
5	vi에서 본문의 변경
6	vi에서 탐색과 교체
7	vi에서의 복사
8	vi에서의 취소

(표계속)

표번호	vi함수
9	본문의 보관과 vi의 탈퇴
10	vi의 선택항목들
11	vi의 상태들
12	vi에서 위치지적과 표식붙이기
13	vi에서 행들의 련결
14	vi에서 유표위치 설정과 화면조절
15	vi의 쉘탈퇴지령들
16	vi의 마크로와 생략부호들
17	vi에서 본문의 들여쓰기화
18	vi의 쉘려파기들
19	vi에서 패턴정합

정규식

정규식은 사용자들이 탐색하는 패턴들을 서술한 기호렬식이다. 정규식에서는 보통 통용기호를 리용한다.

- 정규식은 쉘에서 리용되는 파일정합패턴들과는 다르다. 여기서 고찰하는 정규식은 쉘 및 다른 많은 프로그람들에서 다 리용되는 일반적인것이지만 파일정합패턴은 쉘과 find와 같은 프로그람에서만 리용되는 국부적인것이다.
- 정규식은 웃반점(')안에 써서 쉘의 인수로 리용할수 있다. 웃반점안에 포함된 기호들은 탐색을 조종하는 기호 즉 메타기호로 된다.

기호렬 및 통용기호의 사용

grep 혹은 vi와 같은 프로그람을 사용할 때 사용자가 정규식을 주면 그 프로그람은 그 정규식에 들어 맞는 대상을 탐색한다. 이때 정규식은 하나의 기호렬이거나 통용기호일수 있다.

표 8-2 메타기호와 그 리용프로그람

메타기호	awk	grep	sed	vi	사 용
.	○	○	○	○	임의의 한 기호를 맞춘다.
*	○	○	○	○	*앞에 있는 기호가 임의의 개수만큼 반복되여 있을 때 그 기호렬을 맞춘다.
[…]	○	○	○	○	[…]안에 있는 임의의 한 기호를 맞춘다.

(표계속)

메타기호	awk	grep	sed	vi	사 용
$	○	○	○	○	행의 끝을 맞춘다.
^	○	○	○	○	행의 시작을 맞춘다.
\	○	○	○	○	\뒤에 있는 특수기호를 탈퇴시킨다.
\{n,m}\	○	○	×	×	n과 m사이에 있는 한 기호의 발생령역을 맞춘다.
+	○	×	×	×	이전의 정규식을 한번이상 맞춘다.
?	○	×	×	×	이전의 정규식을 기껏 한번만 맞춘다.
|	○	×	×	×	이전 혹은 이후의 정규식을 들어 맞출수 있다.
()	○	×	×	×	정규식을 표준적인 괄호형식으로 그룹화한다.
\{\}	×	×	×	○	단어의 시작 혹은 끝을 맞춘다.

표 8-2에서는 메타기호와 그것을 리용하는 프로그람들에 대한 목록을 보여 주고 있다.

방식과 표기법

vi를 처음으로 사용하는 사용자들이 부딪치게 되는 특징은 vi가 여러가지 방식들을 가진다는것이다. 여기서는 vi가 가지고 있는 방식들과 그 방식에 기초한 vi의 기본연산들에 대하여 고찰한다.

지령방식(Command Mode)에서는 사용자가 입력하는 모든것이 지령으로 해석된다. 지령방식에서 사용자는 필요한 위치로 유표를 움직이면서 여러가지 작용을 규정할수 있다.

입력방식(Input Mode)에서는 사용자가 입력하는 모든것이 파일에 추가되는 정보로 된다. vi를 기동시키면 표준적으로 지령방식으로 된다. escape건을 눌러 임의의 순간에 입력방식으로부터 지령방식으로 절환할수 있다. 또한 지령방식에서 입력방식의 지령들을 입력하면 지령방식으로부터 입력방식으로 절환할수 있다.

vi지령에는 실지로 표준적인 형식이 없기때문에 표 8-3에서 일반적인 표기법을 주고 있다.

표 8-3에서는 vi의 방식 및 지령들을 개괄하고 있다.

표 8-3 vi의 방식들과 표기법들

방식 혹은 표기법	해 설
지령방식	지령방식에서 사용자는 본문의 입력이나 변경이 아니라 유표의 이동이나 본문의 제거와 같은 지령들을 줄수 있다. 본문의 삽입이나 추가를 위한 i 혹은 a와 같은 입력방식지령들을 주면 입력방식으로 절환할수 있다.
입력방식	한개이상의 본문기호를 삽입하거나 변경시킬수 있다. escape건으로 지령방식으로 절환할수 있다.
：(두점 지령)	：으로 시작하는 지령들은 <Enter>건을 눌러 완성된다.
^(조종지령)	조종건(^)을 누른 상태에서 다른 건을 누른다. 실례로 **^g**는 조종건을 누른 상태에서 g를 누르면 편집하고 있는 파일에 대한 상태를 얻게 된다.
file(파일이름)	많은 지령들은 파일이름을 규정할것을 요구한다. 실례로 지령 **vi** file에서 file에는 편집하려는 파일의 이름을 주어야 한다.
char(기호)	하나의 기호를 요구하는 지령들도 있다. 실례로 **f**char지령에서 char에는 탐색하려는 기호를 주어야 한다.
cursor_command (유표이동지령)	수행시키려는 유표이동지령을 규정하여야 한다. 실례로 **d**cursor_command지령에서 수행시키려는 유표이동지령을 cursor_command에 주어야 한다.
string(기호렬)	기호렬을 지정하여야 하는 지령들도 있다. 실례로 /string에서 탐색하려는 기호렬을 주어야 한다.

vi대화조종의 시작

이제는 실지로 파일을 편집하여 보자. 이 장에서의 거의 모든 실례들은 vi의 여러가지 지령을 사용하고 있으며 X Windows에서 그 결과들을 보여 주고 있다. 아무것이나 실례를 통하여 학습하는것이 가장 좋은 방법이다. 여기서는 많은 실례를 제공할뿐아니라 X Window에서 그 실행결과를 고찰하였기때문에 독자들이 직접 매 지령의 결과를 볼수 있다. 지령행에서 다음과 같이 vi를 입력하고 그뒤에 편집하려는 파일을 입력한다.

$ vi wisdom

그러면 그림 8-1에서 보여 주는것처럼 파일 wisdom을 편집하게 된다.

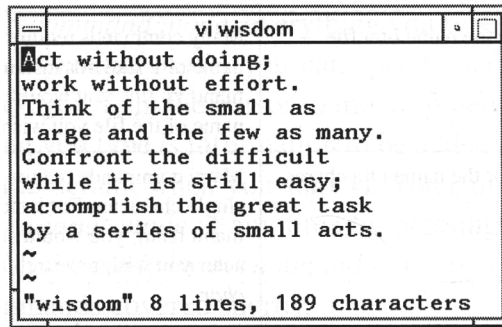



그림 8-1. 파일wisdom의 편집

그림 8-1의 제일 아래행은 vi의 통보행이다. vi를 입장시킨후 통보행은 파일이름과 그 파일의 행개수, 기호개수를 지적한다. 어떤 지령을 주는가에 따라 여러가지 통보가 나타난다. wisdom파일의 통보행우에 있는 2개의 행에 ~가 나타나고 있는것은 행들이 화면을 다 채울만큼 많지 못하기때문이다. 그림 8-1에서 유표는 첫행의 첫 기호에 위치하고 있다.

파일목록을 리용하여 여러개의 파일이름들을 규정할수 있으며 첫번째 파일을 보관한후에 **:n**을 입력하여 두번째 파일로 넘어 가는 방식으로 작업을 계속할수 있다. 또는 하나의 파일을 규정하고 파일의 마지막행에 유표를 지정하여 놓을수 있다. 표준적으로 유표는 그림 8-1에서처럼 파일의 첫 기호에 놓인다.

표 8-4는 vi대화조종을 시작할수 있는 여러가지 방법들을 보여 주고 있다.

표 8-4　　　　vi대화조종의 시작

지　령	해　설
vi file	file을 편집한다.
vi -r file	지령 crash의 수행후에 마지막으로 보관된 file 방안을 편집한다.
vi -R file	file을 읽기전용방식으로 편집한다.
vi +n file	file을 편집하고 유표를 n행에 놓는다.
vi +file	file을 편집하고 유표를 마지막행에 놓는다.
vi file1 file2 file3…	file1 - file3을 편집하고 file1의 변화를 보관한후에 **:n**을 입력하여 file2에로 이행할수 있다.
vi +/string file	file을 편집하고 유표를 string기호렬을 포함하는 행의 시작에 놓는다.

그림 8-2는 wisdom을 편집하면서 유표를 지령 vi +5 wisdom으로 5번째 행에 놓는것을 보여 주고 있다.

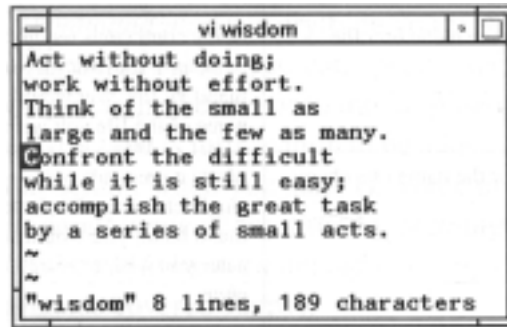



그림 8-2. 지령 vi+5 wisdom에 의하여
5번째행에 유표를 놓고 편집

그림 8-3은 wisdom을 편집하면서 유표를 지령 vi + wisdom으로 파일의 마지막행에 놓는것을 보여 주고 있다.

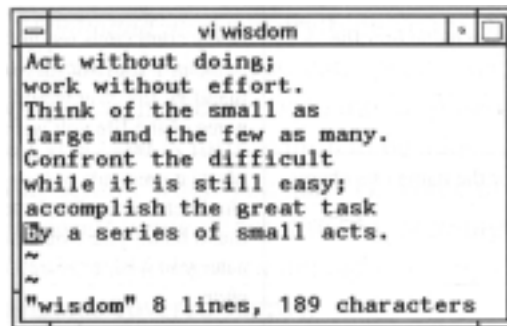



그림 8-3. 지령 vi+ wisdom에 의하여
유표를 마지막행에 놓고 편집

그림 8-4는 wisdom을 편집하면서 유표를 지령 vi +/task wisdom으로 task를 포함하는 행에 놓는것을 보여 주고 있다.

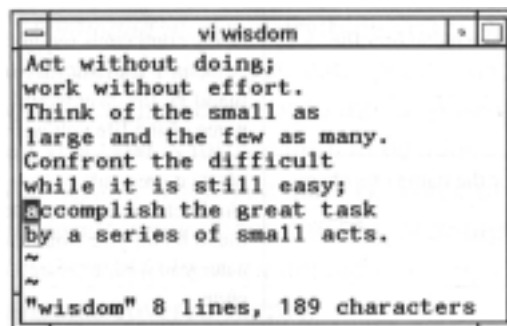



그림 8-4. 지령 vi+/ task wisdom에 의하여
유표를 task를 포함하는 행에 놓고 편집

유표조종지령들

vi에서 건을 다루는 기술은 유표를 해당한 위치에 가져 가는것이다. 사용자는 이러한 기능을 지령방식에서 수행한다. 화면에서 유표를 움직이는 방법에는 여러가지가 있다. 표 8-5에서는 유표이동에 리용되는 공통적인 지령들을 개괄한다.

표 8-5 vi의 유표조종지령들

지 령	유 표 이 동
h 혹은 **^h**	한개 기호 왼쪽에로 이동한다.
j 혹은 **^j** 혹은 **^n**	한개 행 아래에로 이동한다.
k 혹은 **^p**	한개행 우에로 이동한다.
l 혹은 **<Space>**	한개 기호 오른쪽에로 이동한다.
G	파일의 마지막행에로 이동한다.
n**G**	n번 행으로 이동한다.
G$	파일안의 마지막기호에로 이동한다.
1G	파일의 첫행에로 이동한다.
w	다음단어의 시작에로 이동한다.
W	반점을 무시하면서 다음단어의 시작에로 이동한다.
b	앞 단어의 시작에로 이동한다.
B	반점을 무시하면서 앞 단어에로 이동한다.
L	화면의 마지막행에로 이동한다.
M	화면의 중간행에로 이동한다.
H	화면의 첫행에로 이동한다.
e	다음 단어의 끝에로 이동한다.
E	반점을 무시하면서 다음단어의 끝에로 이동한다.
(	문장의 시작에로 이동한다.
)	문장의 끝에로 이동한다.
{	단락의 시작에로 이동한다.
}	다음 단락의 시작에로 이동한다.
0 혹은 **|**	현재행의 첫렬에로 이동한다.
n|	현재행의 n번렬에로 이동한다.
^	현재행의 첫 비공백기호에로 이동한다.
$	현재행의 마지막기호에로 이동한다.
+ 혹은 <Enter>	다음행의 첫기호에로 이동한다.
-	앞행의 첫 비공백기호에로 이동한다.

사용자들은 이런 지령들을 기억하여 유표를 목적하는 위치에 자유롭게 이동시키도록 하여야 한다. 처음에는 이상하게 생각될수 있지만 이것은 vi가 작업하는 방식이기때문에 여기에 익숙하여야 한다. 그림 8-5와 그림 8-6은 wisdom파일에서 이러한 유표이동프로쎄스를 보여 주고 있다. 그림의 왼쪽 부분에서는 지령을 수행하기전상태를 보여 주고 오른쪽 부분에서는 지령수행결과를 보여 주고 있다. 중간에서는 지령들을 보여 주고 있다.

그림 8-5. vi에서 유표이동의 실례

(h, j, k, l)

그림 8-6. vi에서 유표이동의 실례(G, w, b, e)

vi에서 본문의 추가

지금까지는 유표의 이동방법에 대하여 고찰하였다. 왜냐하면 본문을 추가하려면 해당한 위치에 유표를 가져다 놓아야 하기때문에 유표이동을 처음으로 고찰하는것이 필요하였다.

표 8-6에서는 본문을 추가하기 위한 몇가지 지령들을 개괄한다.

표 8-6 vi에서 본문의 추가방법

지 령	삽 입 작 용
a	유표뒤에 새로운 본문을 추가한다.
A	현재행뒤에 새로운 본문을 추가한다.
I	유표앞에 새로운 본문을 삽입한다.
I	현재행의 시작앞에 새로운 본문을 삽입한다.
o	현재행아래에 행을 열고 새로운 본문을 삽입한다.
O	현재행우에 행을 열고 새로운 본문을 삽입한다.
:r file	file을 읽고 그것을 현재행뒤에 삽입한다.
:n**r** file	file을 읽고 그것을 n번째 행뒤에 삽입한다.
escape	지령방식으로 전환한다.
^**v** char	삽입에서 char의 특수한 의미를 무시한다. 이것은 특수기호의 삽입을 위한것이다.

wisdom에 그림 8-7에서처럼 본문을 추가하여 보자.

그림 8-7. vi에서 본문추가의 실레 (a, i, o, O)

vi에서 본문의 제거

표 8-7에서는 본문을 제거하기 위한 몇가지 지령들을 개괄한다.

표 8-7 vi에서 본문의 제거방법

지　령	제 거 작 용
x	유표위치에서 한개 기호를 제거한다.
*n***x**	현재 위치에서 시작하여 n개 기호들을 제거한다.
X	앞의 기호를 제거한다.
*N***X**	앞의 n개 기호를 제거한다.
Dw	다음 단어의 시작까지 제거한다.
*N***dw**	현재위치부터 시작하여 다음의 n개 단어들을 제거한다.
dG	파일의 끝까지 행들을 제거한다.
dd	전체 행을 제거한다.
*n***dd**	현재위치로부터 시작하여 n개 행들을 제거한다.
db	앞단어를 제거한다.
*n***db**	현재위치로부터 시작하여 앞의 n개 단어들을 제거한다.
:*n*,*m***d**	n부터 m번까지의 행들을 제거한다.
D 혹은 **d$**	유표로부터 행끝까지 제거한다.
d*cursor_command*	*cursor_command*까지 본문을 제거한다.
^h 혹은 **<Backspace>**	삽입할 때에 앞의 기호를 제거한다.
^w	삽입할 때에 앞의 단어를 제거한다.

그림 8-8과 그림 8-9에서는 wisdom으로부터 본문을 제거하는 실례를 보여 주고 있다.

그림 8-8. vi에서 본문제거의 실례(x, X, dw, dG)

그림 8-9. vi에서 본문제거의 실례(dd, db)

vi에서 본문의 변경

본문을 삽입도 하여 보았고 제거도 하여 보았기때문에 이제는 본문을 변경하고 싶을 수도 있다. 표 8-8은 본문을 변경하는 몇가지 지령들을 개괄한다.

표 8-8 vi에서 본문의 변경방법

지 령 (지령앞에 수가 있으면 지령을 그 수만큼 반복한다.)	교 체 작 용
rchar	현재기호를 char로 교체한다.
Rtext escape	현재기호들을 escape가 입력될 때까지 text로 교체한다.
stext escape	현재기호를 text로 치환한다.
S 혹은 **cc**text escape	전체 행을 text로 새로운 본문을 삽입한다.
cwtext escape	현재단어를 text로 변경시킨다.
Ctext escape	현재행의 나머지부분을 text로 변경시킨다.
cG escape	파일의 끝에로 변경시킨다.
ccursor_cmd **text** escape	현재위치로부터 cursor_cmd까지를 text로 변경시킨다.

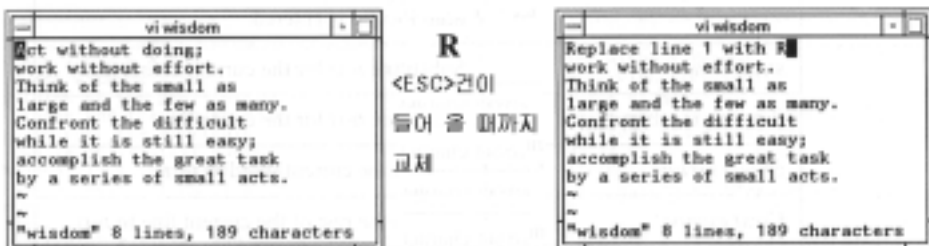
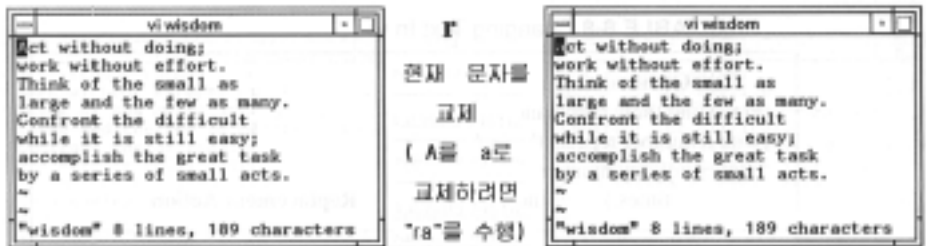
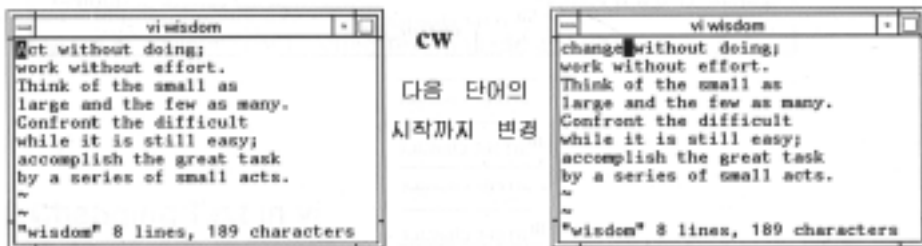
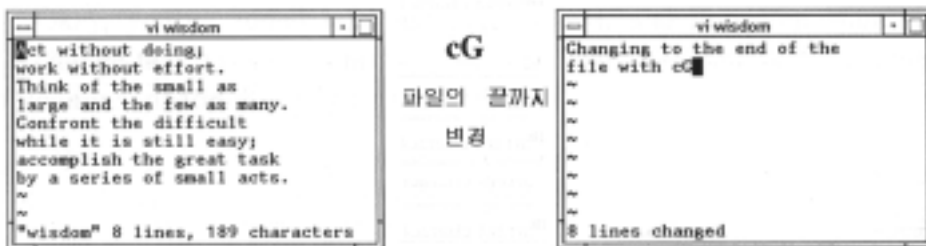
그림 8-10과 그림 8-11에서는 wisdom으로부터의 본문을 교체하는 실례를 보여 주고 있다.

지령실례 : RReplace line 1 with R<ESC>

지령실례 : cwChange<ESC>

지령실례 : cGChanging to the end of the file with cG<ESC>

그림 8-10. vi에서 본문변경의 실례(r, R, cw, cG)

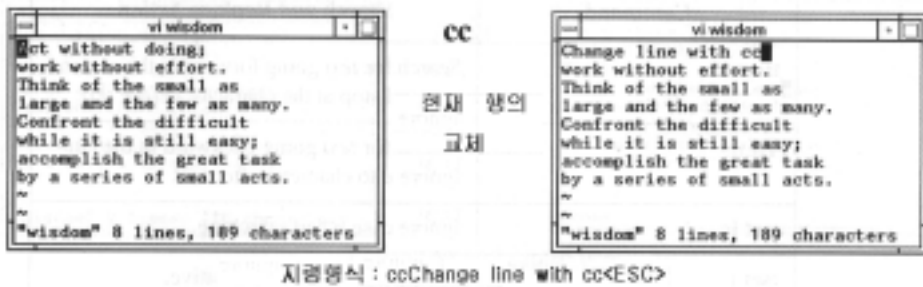



그림 8-11. vi에서 지령 cc에 의한 본문변경의 실례

vi에서의 찾기와 교체

vi에는 많은 찾기와 교체기능이 있다. 표 8-9는 vi의 찾기 및 교체기능들에 대하여 개괄한다.

표 8-9 vi에서의 찾기와 교체

지　　령	찾기 및 교체작용
/text	파일안에서 전진하는 방향으로 text를 찾는다.
?text	파일안에서 후진하는 방향으로 text를 찾는다.
n	초기의 탐색과 같은 방향으로 탐색을 반복한다.
N	초기의 탐색과 반대방향으로 탐색을 반복한다.
ftext	현재행에서 전진하는방향으로 text를 찾는다.
Ftext	현재행에서 후진하는방향으로 text를 찾는다.
ttext	현재행에서 전진하는 방향으로 text를 탐색하고 text앞의 기호에서 정지한다.
Ttext	현재행에서 후진하는 방향으로 text를 탐색하고 text뒤의 기호에서 정지한다.
:set ic	탐색할 때 대소문자차이를 무시한다.
:set noic	탐색할 때 대소문자를 구별한다.
:s/oldetxt/newtext/	oldtext를 newtext로 교체한다.
:m,n**s**/oldetxt/newtext/	m~n행사이에서 oldtext를 newtext로 교체한다.
&	마지막 **:s**지령을 반복한다.
:g/text1/**s**/text2/text3	text1을 포함하는 행을 찾고 text2를 text3으로 교체한다.
:g/text/command	text를 포함하는 모든 행들에 command를 실행시킨다.
:v/text/command	text를 포함하지 않는 모든 행들에 command를 실행시킨다.

그림 8-12에서는 wisdom에서 본문을 탐색 및 교체하고 있다는것을 보여 주고 있다.

그림 8-12. vi gb에서 탐색과 교체의 실례

:g와 :v에 의하여 발전된 탐색들을 수행시킬수 있다. 아래에서와 같이 while을 포함하는 파일안의 모든 행들을 현시하려면 다음과 같이 하여야 한다.

```
:g/while/p
```

이 지령행에서 /p는 ex편집기와 함께 사용되는 출력지령이다. while을 포함하는 모든 행들을 찾고 그 행들을 제거하려면 다음과 같이 한다.

```
:g/while/d
```

탐색하려는 행번호도 규정할수 있다. while을 포함하는 10부터 20사이에 있는 모든 행들을 찾고 while이 나타나는 행번호를 출력하려면 다음과 같이 하여야 한다.

```
:10, 20g/while/nu
```

:g는 찾으려는 본문을 포함하는 행들에 대하여 지령을 실행하며 :v는 규정된 본문을 포함하지 않는 행들에 대하여 지령을 실행한다. 다음의 3개의 지령은 앞의 3개의 지령과 동일하지만 모두 while을 포함하지 않는 행들에 대하여 수행된다.

:v/while/p

:v/while/d

:10,20v/while/nu

첫번째 지령은 while을 포함하지 않는 행들을 출력한다. 두번째 지령은 while이 나타나지 않는 행들을 제거한다. 세번째 지령은 while이 나타나지 않는 10부터 20사이의 행번호를 출력한다.

vi에서 본문의 복사

표 8-10에서는 본문을 복사하기 위한 몇가지 지령들을 보여 주고 있다.

표 8-10 vi에서 복사방법

지 령	탐색 및 교체작용
yy	현재행을 뽑는다.
nyy	n개의 행들을 뽑는다.
p (소문자)	뽑은 본문을 유표뒤에 놓는다.
P (대문자)	뽑은 본문을 유표앞에 놓는다.
"(a~z)n**yy**	n개의 행을 괄호안에서 규정된 이름을 가진 완충기에 복사한다. n이 생략되면 현재행이다.
"(a~z)n**dd**	괄호안에서 규정된 이름을 가진 완충기에서 n개의 행을 제거한다. n이 생략되면 현재행이다.
"(a~z)**p**	괄호안에서 규정된 이름을 가진 완충기에 있는 행들을 현재행뒤에 놓는다.
"(a~z)**P**	괄호안에서 규정된 이름을 가진 완충기에 있는 행들을 현재행앞에 놓는다.

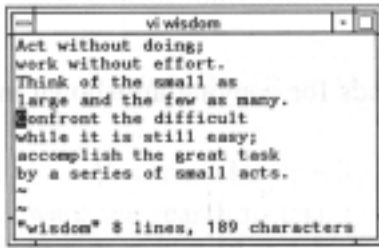
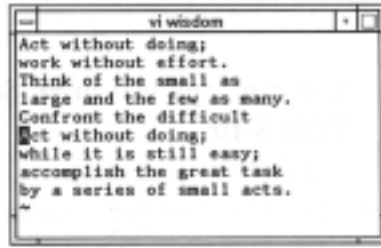
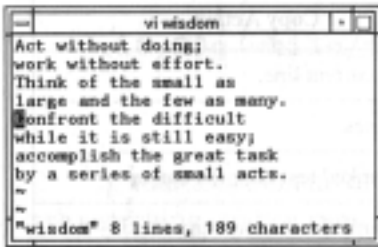
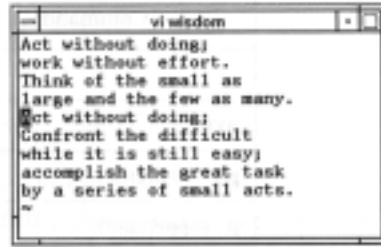
그림 8-13에서는 wisdom안에 있는 본문을 복사하는 실례를 보여 주고 있다.

지령실례 : yy(첫 행에서), 그 다음 p(소문자)

지령실례 : yy(첫 행에서), 그 다음 P(대문자)

그림 8-13. vi에서의 복사

vi에서 취소와 반복

표 8-11에서는 vi에서 취소 및 반복을 진행하는 명령문들을 보여 주고 있다.

표 8-11 vi에서의 취소방법

지 령	취 소 작 용
u	마지막변경을 취소한다.
U	현재행에 대한 모든 변경을 취소한다.
.	마지막변경을 반복한다.
,	거꿀방향으로 **f, F, t, T**탐색지령을 반복한다.
;	마지막 **f, F, t, T**지령을 반복한다.
"np	n번째 제거를 회복한다(제한된 제거회수는 완충기에 들어 있는데 보통 9이다.).
n	마지막 혹은 **?** 탐색지령을 반복한다.
N	거꿀방향으로 마지막 혹은 **?** 탐색지령을 반복한다.

vi에서 본문의 보관과 탈퇴

표 8-12에서는 본문을 보관하고 vi를 탈퇴하는 방법들을 보여 주고 있다.

표 8-12 vi에서의 본문보관과 탈퇴하기

지 령	보관과 탈퇴작용
:w	파일을 보관하지만 vi를 탈퇴하지 않는다.
:w file	file안에서의 변경들을 보관하지만 vi를 탈퇴하지는 않는다.
:wq 혹은 **ZZ** 혹은 **:x**	파일을 보관하고 vi를 탈퇴한다.
:q!	파일을 보관함이 없이 vi를 탈퇴한다.
:e!	마지막쓰기로부터의 변경들을 취소하면서 파일을 재편집한다.

vi의 선택항목

vi에는 설정되거나 설정되지 않는 선택항목들이 많다. 선택항목을 설정하려면 :set option을 입력하고 설정을 해제하려면 :set nooption을 입력한다. 표 8-13은 공통적으로 사용되는 선택항목들을 개괄한다.

표 8-13 vi의 선택항목들

선택항목	작 용
:set all	모든 선택항목들을 출력한다.
:set nooption	option을 해제한다.
:set nu	행번호를 행앞에 붙인다.
:set showmode	입력방식 혹은 교체방식을 보여 주고 있다.
:set noic	탐색할 때 대소문자를 무시한다.
:set list	태브들(^I)과 행끝($)을 보여 준다.
:set ts=8	본문입력에 대하여 태브위치들을 설정한다.
:set window=n	본문창문안에 있는 행들의 개수를 n으로 설정한다.

그림 8-14에서는 행앞에 행번호들을 붙인 상태에서 입력(혹은 교체)방식을 보여 주고 있다.

그림 8-14. vi에서의 선택항목들

표 8-13에서 제시한것외에도 많은 선택항목들이 있다. 아래에서는 지령 :set all에 의하여 UNIX체계에서 생성된 선택항목들의 목록을 보여 주고 있다. 이 지령은 vi에서 가능한 선택항목들을 보려고 할 때 주어야 한다.

: set all

noautoindent
autoprint
noautowrite
nobeautify
directory=/var/tmp
nodoubleescape
noedcompatible
noerrorbells
noexrc
flash
hardtabs=8
noignorecase
keyboardedit
nokeyboardedit!
nolisp
nolist

magic
mesg
nomodelines
nonumber
nonovice
nooptimize
paragraphs=IPLPPPQPP LIpplpipnpbp
prompt
noreadonly
redraw
remap
report=5
scroll=11
sections=NHSHH   HUuhsh+c
shell=/sbin/sh
shiftwidth=8
noshowmatch
noshowmode
noslowopen
tabstop=8
taglength=0
tags=tags /usr/lib/tags
tagstack
term=hp
noterse
timeout
timeoutlen=500
ttytype=hp
warn
window=23
wrapscan
wrapmargin=0
nowriteany
noshowmatch
noshowmode

"no"가 앞에 붙은 선택항목들은 설정되지 않는다는것을 의미한다.

vi의 상태

vi에 있는 지령들을 통하여 많은 상태정보를 얻을수 있다. 현재 행번호, 파일의 행개수, 파일이름 기타 상태정보를 표 8-14에서 보여 준 지령들로 현시할수 있다.

표 8-14 vi의 상태들

선택항목	작 용
:. =	현재 행번호를 현시한다.
:=	파일안에 있는 행개수를 현시한다.
^g	파일이름, 현재 행번호, 파일안에서 총 행개수, 파일위치의 퍼센트값을 현시한다.
:l	태브와 행바꾸기기호와 같은 여러가지 특수기호들을 현시한다.

본문에서의 위치지적과 표식붙이기

본문에서 어떤 부분을 기호표식들로 표시해 놓고 거기로 이동하게 할수 있다. 본문의 이런 부분을 **구획**(Section)이라고 부른다. 표 8-15에서는 vi에서 위치지적과 표식붙이기를 개괄한다.

표 8-15 vi에서 위치지적과 표식붙이기

선택항목	작 용
{	구획을 정의하는 첫렬에 { 를 삽입한다.
[[	구획의 시작으로 보낸다.
]]	다음구획의 시작으로 보낸다.
m(a-z)	현재 위치에 표식 z를 붙이기 위하여 **mz**라고 지령을 준다.
'(a-z)	지령 'z에 의하여 표식 z에로 유표를 이동시킨다.

vi에서 행들의 련결

vi에서 한개이상의 행들을 표 8-16에서 보여 준 지령들에 의하여 련결할수 있다.

표 8-16 vi에서 행들의 련결방법

선택항목	작 용
J	현재행의 끝에 다음행을 련결시킨다.
n**J**	다음에 있는 n개 행들을 련결시킨다.

vi에서 유표위치와 화면조절

표 8-17에서 보여 준 지령들을 리용하여 여러가지 방법으로 파일의 임의의 위치에 유표를 놓을수 있으며 화면을 조절할수 있다.

표 8-17 vi에서 유표위치와 화면조절방법

선택항목	작 용
H	유표를 화면의 제일 웃행에로 이동시킨다.
N**H**	유표를 화면의 제일 웃행으로부터 n번째 행에로 이동시킨다.
M	유표를 화면의 중간에로 이동시킨다.
L	유표를 화면의 제일 아래행에로 이동시킨다.
N**L**	유표를 화면의 제일 아래행으로부터 n번째행에 이동시킨다.
^**e**	화면을 한개 행우에로 움직인다.
^**y**	화면을 한개 행아래로 움직인다.
^**u**	화면을 반페지우로 이동시킨다.
^**d**	화면을 반페지아래로 이동시킨다.
^**b**	화면을 한페지우로 이동시킨다.
^**f**	화면을 한페지 아래에로 이동시킨다.
^**l**	화면을 다시 그린다.
z - <Enter>	현재행을 화면의 꼭대기에 설정한다.
n**z** - <Enter>	행 n을 화면의 꼭대기에 설정한다.
z.	현재행을 중간행에 설정한다.
n**z.**	행 n을 화면의 중간행으로 설정한다.
z -	현재행을 제일 아래행으로 설정한다.
n**z** -	행 n을 화면의 제일 아래에 설정한다.

쉘탈퇴지령

쉘탈퇴지령들을 리용하면 vi에서 탈퇴하지 않고 UNIX지령들을 수행시킬수 있다. 우선 지령 :sh을 써서 간단하게 부분쉘을 시동시킬수 있으며 vi를 끝내지 않고도 편집하고 있는 파일밖에서 지령들을 수행시킬수 있다.

표 8-18에서는 쉘탈퇴지령들을 보여 주고 있다.

표 8-18 vi의 쉘랄되지령들

선택항목	작 용
:!command	:!ls와 같은 쉘지령 command를 수행한다.
:!!	마지막쉘지령을 수행한다,
:r! command	command로부터의 출력을 읽고 삽입한다.
:w! command	현재 편집된 파일을 표준입력으로서 command에 보내며 그 command를 수행한다.
:cd directory	현재 등록부를 directory로 변경한다.
:sh	부분쉘을 시동하고 ^d를 리용하여 vi에로 귀환한다.
:so file	쉘프로그람 file안에 있는 지령들을 읽고 수행한다.

다음의 실례에서는 wisdom파일을 표준입력장치로부터 grep로 보내여 all을 포함하는 모든 행들을 찾도록 지령 :w를 사용하고 있다.

:w ! grep all
Think of the small as
by a series of small acts.

다음의 실례에서는 vi에서 파일 file_with_commands에 있는 지령들을 읽어 수행시키는 지령 :so를 사용하고 있다.

:so file_with_commands

이 파일은 다음과 같은 두개의 지령을 포함하고 있다.

:set nu
:g/all/p

앞의 지령 :so를 줄 때 지령 :set nu에 의하여 행번호들이 나타나며 all을 포함하는 다음의 행들이 출력된다.

Think of the small as
by a series of small acts.

마크로와 생략부호

vi지령들은 개별적으로만이 아니라 몇개의 지령들로 이루어 진 지령묶음렬을 정의하고 거기에 열쇠를 대응시켜 호출하여 쓸수도 있다. 이러한 **마크로**(Macro)를 위한 열쇠를 정의할 때에 기능건과 조종건들, K V g q v * =들은 사용할수 없다. 표 8-19에서는 마크로와 생략부호들을 보여 주고 있다.

표 8-19 vi에서 마크로와 생략부호들

선택항목	작 용
:map key command_seq	command_seq를 수행시키기 위한 key를 정의한다.
:map	상태행에 정의된 마크로전부를 현시한다.
:umap key	key에 대한 마크로를 제거한다.
:ab string1 string2	string1이 삽입될 때에 그것을 string2로 바꾸는 생략부호를 정의한다. string1을 입력할 때에 escape건을 누르면 string2가 삽입된다.
:ab	모든 생략부호들을 현시한다.
:cd directory	현재작업등록부를 directrory로 변경한다.
:una string	string생략부호를 해제한다.
	조종건, 기호, 기능건, K V g q v * =는 사용하지 못한다.

다음의 실례에서는 지령 map에 의하여 유표가 끝에 갈 때에 자동적으로 본문을 추가할수 있게 하고 있다.

:map e ea

이 지령은 e를 ea에 대응시킨다. 유표가 e로 다음 단어의 끝으로 갈 때에 ea에 의하여 입력방식으로 놓이게 되며 따라서 단어의 끝에 새 본문을 즉시에 추가할수 있게 된다.

ab지령으로 긴 렬을 생략할수도 있다. 실례로 system administration을 다음의 지령에 의하여 sa로 생략할수 있다.

:**ab sa system administration**

이 지령이 수행된후부터 본문을 삽입할 때마다 sa를 입력하고 escape건을 누르면 system administration이 나타난다. 즉 sa는 system administration의 생략부호이다.

본문의 들여쓰기

사용자들은 여러가지 방법으로 본문을 **들여쓰기**(Indent)할수 있다. 표 8-20에서는 공통적으로 사용되는 본문의 들여쓰기지령들을 보여 주고 있다.

이행너비를 조절하기전에 :set all지령을 리용하여 이행너비로 설정되는 현재의 기호개수를 볼수 있다. 보통 기정값은 8개 기호이다. 16개 기호로 설정하려면 다음과 같이 한다.

:set sw=16

표 8-20 vi에서 본문의 들여쓰기

선택항목	작 용
^**i** 혹은 태브	본문을 삽입할 때에 이행너비만큼 삽입된다. 즉 이행너비가 정의된다.
:set ai	자동들여쓰기를 설정한다.
:set sw=n	이행너비를 n개 기호로서 설정한다.
n<<	n개 행을 하나의 이행너비만큼 왼쪽으로 이동시킨다.
n>>	n개 행을 하나의 이행너비만큼 오른쪽으로 이동시킨다.

다음의 지령으로 아래의 3개 행을 오른쪽으로 16개 기호만큼 이행할수 있다.

3>>

쉘려과기

편집하고 있는 파일로부터의 정보를 어떤 지령에 보내여 그 지령의 결과로 초기본문을 바꿀수 있다. 표 8-21은 쉘려과기를 보여 주고 있다.

표 8-21 쉘려과기들

선 택 인 수	작 용
!cursor_command command	현재 위치로부터 cursor_command에 의하여 규정된 위치까지의 본문을 쉘의 command에 보낸다. 실례로 !}grep admin은 현재 위치로부터 단락의 끝까지에 있는 본문을 취하여 그 본문에 단어 admin을 찾기 위한 grep를 수행시키며 grep의 결과로 본문을 바꾼다.

패턴정합

패턴정합(pattern matching)은 편집하고 있는 파일안에서 패턴들을 찾을수 있게 하여준다. 그다음에는 찾은 자료에 대하여 변경과 같은 여러가지 작용을 할수 있다. 표 8-22에서는 아주 일반적인 패턴정합지령들을 보여 주고 있다.

표 8-22 vi에서의 패턴정합

선택항목	작　용
^(caret)	행의 시작을 맞춘다. 실례로 행의 시작에 있는 Think를 탐색하려면 다음과 같이 하여야 한다. **/^Think** 행끝에 맞추려면 $를 쓸수 있다. 행끝을 맞추고 모든 빈 행들을 제거하려면 다음과 같이 하여야 한다. **:g/^$/d**
$	행끝을 맞춘다. 실례로 last.를 행바꾸기기호뒤에 올 때에만 맞추려면 다음과 같이 하여야 한다. **/last. $**
.	임의의 한개 기호를 맞춘다.
\<	단어의 시작을 맞춘다.
\>	단어의 끝을 맞춘다.
[string]	string안에 있는 임의의 한개 기호를 맞춘다. **mp, mP, Mp, MP**를 찾으려면 다음과 같이 하여야 한다. **/[nM][pP]** **input** 혹은 **Input**나 **INPUT**의 모든 경우를 변경시키려면 다음과 같이 하여야 한다. **:%s/[Ii]nput/INPUT/g**
[^string]	string안에 없는 임의의 기호를 맞춘다.
[a-p]	a부터 p까지의 임의의 기호를 맞춘다.
*	식안에서 이전기호의 출현을 한번이상 맞춘다.
\	다음기호에 대한 탈퇴를 의미한다. 실례로 [를 탐색하려면 다음과 같이 하여야 한다. **/ \ [**
\\	\기호를 탈퇴시킨다.

처음으로 패턴정합을 할 때 약간 혼돈될수도 있다. 그리하여 아래에서는 몇가지 단순한 실례들을 통하여 패턴정합을 익숙하도록 한다. 이러한 패턴정합수법들은 vi의 밖에서도 할수 있는것들이다.

모임의 정합

[ ]연산자로서 m, f, p 등의 임의의 한개 기호를 맞추려면 다음과 같이 하여야 한다.

```
/[mfp]
```

첫 문자의 대소를 구별하여 단어를 맞추어 보자. 즉 input 혹은 Input를 맞추려면 다음과 같이 하여야 한다.

/[Ii]nput

input 혹은 Input를 맞춘후에 그것을 INPUT로 변경하려면 다음과 같이 하여야 한다.

:%s/[Ii]nput/INPUT/g

한개 기호이상을 탐색하는 식을 다음과 같이 쓸수 있다. 즉 mp, mP, Mp, MP를 맞추려면 다음과 같이 하여야 한다.

/[mM] [pP]

령역정합

[ ]연산자를 리용하여 어떤 령역안에 있는 한개의 기호에 대한 정합도 할수 있다. 실례로 파일에서 임의의 수자들의 발생을 찾으려고 하면 다음과 같이 하여야 한다.

/[0123456789]

혹은

/[0-9]

대문자와 소문자를 고려한 임의의 문자를 찾으려면 다음과 같이 하여야 한다.

/[a-zA-Z]

특수의미를 가지는 기호들을 탐색하려면 특수기호앞에 \을 놓아야 한다.
실례로 특수기호 [를 탐색하려면 다음과 같이 하여야 한다.

/ \[

이 지령은 처음으로 나타난 [을 탐색한다.

행탐색의 시작과 끝

행의 시작과 끝에서만 발생하는 패턴들을 정합할수 있다. 행의 시작에서 정합을 하려면 목적하는 패턴앞에 ^을 놓는다. 실례로 행의 시작에서 나타나는 Think들만 맞추려면 다음과 같이 하여야 한다.

/^Think

행의 끝에서 패턴정합을 하려면 목적하는 패턴뒤에 $를 놓는다. 실례로 행의 끝에서 나타나는 last.들만 맞추려면 다음과 같이 하여야 한다.

/last. $

이 지령은 last.을 그뒤에 새행이 올 때에만 들어 맞춘다.

지령소개

vi

vi – 시각적인 편집기를 실행시킨다.

---

vi(1)

이름

vi, view, vedit – 화면지향의(시각적인) 본문편집기이다.

형식

vi [-] [-l] [-r] [-R] [-t tag] [-v] [-V] [-w size] [-x] [+command] [file …]

XPG4에서의 형식

vi [-rR] [-c command] [-t tag] [-w size] [file …]

절대형식

vi [-rR] [+command] [-t tag] [-w size] [file …]
view [-] [-l] [-r] [-R] [-t tag] [-v] [-V] [-w size] [-x] [+command] [file …]
vedit [-] [-l] [-r] [-R] [-t tag] [-v] [-V] [-w size] [-x] [+command] [file …]

주의

ex, edit, vi, view, vedit와 같은 프로그람이름들은 다 동일한 프로그람에 대한 개별적인 이름이다. 이 지령소개에서는 vi/ view/ vedit에 있는 기능들을 설명한다.

해설

vi(시각적인)프로그람은 행편집기 ex에 기초하고 있는 화면지향의 본문편집기이다(ex(1)을 참고). vi안에서도 ex에로 절환할수 있다. 행편집기의 지령들과 행편집기의 선택항목들은 ex에 서술되여 있다. 여기서는 시각적인 방식들만 서술한다.
view프로그람은 읽기전용선택항목이 설정된다는것을 제외하고는 vi와 동등하다(ex(1)을 참고).

vedit프로그람은 초학자들에게 친절한것이다. 보고서편집기선택항목이 1로 설정되여 있으며 nomagic, novice, showmode선택항목들도 설정되여 있다.

vi에서 말단화면은 편집되고 있는 파일의 기억기복사에 대한 창문으로 작용한다. 복사파일에 대한 변경내용은 화면표시장치에 반영된다. 화면에서 유표의 위치는 복사파일안에서의 위치를 지적한다.

환경변수 TERM은 terminfo자료기지에서 정의되는 말단의 종류를 규정하여야 한다. 그밖의 경우에는 통보가 현시되며 행편집기가 입장한다.

ex에서와 같이 편집기초기설정스크립트들은 환경변수 EXINIT나 현재 혹은 홈등록부에 있는 . exrc파일안에 놓일수 있다.

선택항목과 인수들

vi는 다음과 같은 지령행선택항목들과 인수들을 인식한다.

- `-` 모든 사용자들로부터의 입력을 금지한다. 편집기지령들은 스크립트로만 취해 진다.
- `-l` lisp편집선택항목을 설정한다(ex(1)을 참고). 즉 lisp코드에 적중한 들여쓰기들을 제공한다. vi에서의 (,), {, }, [[,]] 지령들은 lisp원천코드에 대한 기능에 의하여 수정된다.
- `-r` 편집기 혹은 체계의 고장후에 규정된 파일들을 되살려 준다. 파일이 주어 지지 않으면 보관된 모든 파일의 목록이 출력된다. 보관된 파일을 되살리려면 사용자는 그 파일의 소유자가 되여야 한다. 상급사용자는 다른 사용자들이 소유한 파일들을 되살릴수 없다.
- `-R` 읽기전용선택항목을 설정하여 파일을 중복하여 쓰는것을 막는다.
- `-t tag` tag지령을 수행하여 미리 정의된 파일을 적재하고 가리킨다(ex(1)에서 tag지령과 tag편집기선택항목들을 참고).
- `-v` 시각적방식(vi)을 입장시킨다. ex에서만 유효하며 vi에서는 아무런 효과도 없다.
- `-wsize` 창문편집기선택항목의 값을 size에 설정한다. 만일 size가 생략되면 기정값은 3이다.
- `-x` 암호화방식을 결정한다. 사용자는 이 방식에서 보호된 파일의 창조나 편집을 허용하는 열쇠를 입력하여야 한다(ex(1)에서 crypt지령을 참고).
- `-c command` XPG4에서만 가능하다.

+command

규정된 ex지령방식의 지령들을 수행시켜 편집을 시작한다. 표준적인 ex지령행구체례에서와 마찬가지로 command는 | 에 의하여 분리된 여러개의 ex지령들로 구성된다. 이 방식에서 입력방식을 입장시키는 지령들을 사용하면 예견치 않은 결과가 나올수 있다.

file

편집하여야 할 한개 혹은 여러개의 파일들을 규정한다. 여러개의 파일이 규정되면 주어 진 순서대로 처리된다. 선택항목 -r가 규정되면 거꿀순서로 처리된다.

XPG4에서만 가능한것으로서 -t tag와 -c command 둘 다 주어 지면 -t tag가 먼저 처리된다. 즉 -t로 설정된 tag를 포함하는 파일이 먼저 처리되고 그다음 지령이 수행된다.
vi가 기동되면 지령방식으로 된다. 입력방식은 본문을 삽입하거나 변경시킬 때에 리용하는 몇가지 지령들에 의하여 초기화된다.

입력방식에서 escape기호는 입력방식을 탈퇴하는데 쓰이지만 련속적인 두개의 escape기호들은 doublespace선택항목이 설정되는 경우에 입력방식을 탈퇴하도록 한다.
지령방식에서 escape는 부분적인 지령을 취소하는데 리용된다. 만일 편집기가 입력방식이 아니고 부분적으로 입력된 지령이 하나도 없다면 말단종이 울린다.

경고
escape는 제일 아래의 행지령을 완성한다(아래를 참고).

화면의 제일 마지막행은 탐색지령(/과 ?), ex지령들(:), 체계지령들(!)에 대한 입력을 재촉하는데 리용된다. 또한 오유를 통보하거나 다른 통보들을 출력하기 위하여 리용된다.

본문의 입력 혹은 마지막행에 지령들을 입력하는 동안에 SIGINT의 접수는 입력을 끝내거나(혹은 지령을 취소) 편집기의 지령방식으로 귀환한다. 지령방식일 때 SIGINT는 경고를 울린다. 일반적으로 경고는 정의되지 않은 열쇠를 리용하는것과 같은 오유를 지적한다.

화면에서 ~만으로 이루어 진 행은 그 행우에 있는 행이 파일의 마지막행이라는것을 가리킨다. 제한된 국부적지능을 가진 말단들은 @가 붙은 행들을 화면에 현시하게 한다. 즉 이것들은 파일의 행에 대응하지 않는 화면의 공간을 가리킨다(이러한 행들은 ^R에 의하여 제거될수 있으며 편집기가 이런 행들이 없이 화면을 재설정하도록 한다.).
만일 체계가 고장나고 vi가 내부적인 오유나 뜻밖의 신호에 의하여 억제 당했다면 vi는 보관시키지 않은 변경내용들이 있는 경우에 그 변경을 완충기에 보관하게 된다. 선택항목 -r는 보관된 변경내용들을 탐색하게 한다.

vi본문편집기는 SIGWINCH신호를 보장하며 창문크기의 변경에 따라 화면을 다시 현시한다.

지령개요

대부분의 지령들은 제일 앞에 하나의 수를 인수로서 접수한다. 그 번호는 크기 혹은 위치(화면현시 혹은 이행지령인 경우) 또는 반복개수(본문을 변경하는 지령인 경우)이다. 간단히 고찰하기 위하여 이 선택적인 인수는 그것의 효과가 서술될 때 고찰하기로 한다.

c, d, y, <, >, !, = 연산자들은 이행지령뒤에 놓이며 본문에 영향을 미치게 한다. 규정된 령역은 현재 유표위치에서 시작하고 이행된 유표위치앞에서 끝나는 령역이다. 만일 지령이 행들에만 작용한다면 이 령역안에 부분적으로 혹은 전면적으로 떨어 지는 모든 행들이 영향을 받는다. 그밖의 경우에는 정확히 표식된 령역만 영향을 받는다.

아래의 서술에서 조종기호들은 ^X형식(Ctrl - X)으로 지적된다. 공백기호는 <Space>, <Tab>, <Alt>기호를 의미한다. <Alt>기호는 langinfo(5)에서 서술되는 ALT_PUNCT항목의 첫 기호이다.

다른 방법으로 규정되지 않는 한 지령들은 지령방식에서 해석되며 입력방식에는 아무런 영향도 주지 않는다.

^B 본문의 앞창문을 현시하도록 후진하는 방향으로 흘러 보낸다. 앞에 있는 수자는 흘러 보내는 창문의 개수를 규정한다. 가능하면 두개의 행은 겹쳐 진다.

^D 본문을 창문의 절반만큼 전진하는 방향으로 흘러 보낸다. 앞의 수는 흘러 가는 론리적행들의 개수이며 앞으로의 ^D와 ^U지령을 위하여 기억된다.

^D(입력방식)
자동들여쓰기 혹은 ^T에 의하여 제공된 들여쓰기를 다음의 이행너비공백들에 예비복사한다. 행의 시작이 아닌 다른데서 ^T에 의하여 삽입된 공백들은 ^D로서 예비복사되지 않는다. ^은 현재입력방식의 현재 및 다음입력행들에 대한 들여쓰기를 모두 제거한다. 이 제거는 새로운 들여쓰기가 삽입될 때까지 계속된다.

^E 유표는 가능한껏 남겨 두면서 한개 행을 전진하는 방향으로 흘러 보낸다.

^F　　본문의 창문을 다음 창문의 현재위치까지 전진하는 방향으로 흘러 보낸다. 앞의 수는 흘러 보내는 창문의 개수를 규정한다.

현재행이 현시되고 유표는 현재행의 비공백기호 혹은 공백행인 경우 첫 기호로 이행된다(XPG4에서만 가능).

^G　　현재 파일이름과 행개수, 현재위치 등의 정보를 출력한다(ex의 f 지령과 동등하다.).

^H　　왼쪽으로 한개의 공백을 이동한다. 앞의 수는 이동하는 공백의 개수를 규정한다.

^H(입력방식)

이미 입력된 기호를 화면에서 지우지 않고 유표를 왼쪽으로 한 기호 이행시킨다. 그 기호는 보관된 본문에서 제거된다.

^J　　유표를 동일한 렬에서 한개 행아래로 이동시킨다. 앞의 수는 아래로 이행시키는 행의 개수를 규정한다. ^N, j와 동등하다.

^L　　화면을 지우고 다시 그린다.

^M　　다음행의 첫 비공백기호로 이동한다. 앞의 수자는 이동하는 행 개수를 규정한다.

^N　　^J, j와 동등하다.

^P　　유표를 동일한 렬에서 한개 행우로 이동시킨다. 앞의 수는 우로 이동시키는 행의 개수를 규정한다. k와 동등하다.

^R　　@로 표식된 거짓행들을 지우면서 현재화면을 다시 그린다.

^T　　tag탄창(ex(1)의 pop지령을 참고).

^T(입력방식)

이행너비만큼 공백을 삽입한다. 만일 행시작이라면 삽입된 공백은 ^D를 통하여 예비복사될수 있다.

^U　　본문의 창문절반을 우로 흘러 보낸다. 앞의 수는 우로 흘러 보내는 행들의 개수를 규정한다. 전진하는 방향으로의 ^D와 ^U지령을 위하여 기억된다.

^V　　입력방식에서 escape를 포함한 특수기호의 삽입을 위하여 다음 기호를 " "안에 넣도록 한다.

^W　　입력방식에서 한개 단어만큼 후진하는 방향으로 이행한다. 제거된 기호들은 화면에 남아 있는다.

^Y　　유표는 그대로 두면서 한개 행만큼 후진하는 방향으로 흘러 보낸다.

^[부분적으로 형성된 지령을 취소한다. 그런 지령이 없으면 경고를 울린다.

입력방식에서 ^[는 입력방식을 끝낸다. 그러나 두개의 련속적인 escape기호들은 두배공백편집기선택항목이 설정되여 있는 경우에 입력방식을 끝내도록 요구한다(ex(1)을 참고).

화면의 제일 아래행에 지령을 입력할 때 입력을 끝내고 지령을 실행한다.

^\ vi를 끝내고 ex지령방식을 입장시킨다. 만일 입력방식이면 먼저 입력을 끝낸다.

^] 유표위치에서 두개의 단어를 취하여 tagMbobc편집기지령을 실행한다(ex(1)을 참고).

^^ 이전 파일로 귀환한다(:ex #지령과 동등하다.).

space 한개의 공백을 오른쪽으로 이동시킨다. 앞의 수는 이동하는 공백의 개수를 규정한다(l과 동등하다.).

erase 사용자가 정의한 제거기호를 규정한다(stty(1)을 참고). ^H와 동등하다.

kill 사용자가 정의한 무시기호를 규정한다(stty(1)을 참고). 입력방식에서 kill은 행을 지우지 않고 현재 입력행의 시작까지 후진하는 방향으로 갈수 있다.

susp 편집대화조종을 정지(보류)하고 쉘호출에로 귀환한다. susp는 사용자가 정의한 과제조종보류기호이다(stty(1)을 참고하고 보류편집기지령들에 대하여 ex(1)을 참고).

! 주어 진 행들을 표준입력장치의 완충기로부터 규정된 체계지령에 통과시키고 그 지령으로부터 표준출력장치로 주어 진 행들을 교체하는 연산자이다. ! 는 통과시킬 행들을 규정하는 이행지령뒤에 놓인다. 앞의 수는 ! 뒤에 있는 이행지령에 통과된다.

!! 과 개수앞에 놓인 ! 는 현재행부터 시작하여 여러개의 행들이 통과되게 한다.

" 완충기의 이름을 규정한다. 완충기들은 1부터 9까지 이름이 불을수 있으며 편집기들은 거기에 제거된 본문을 넣는다. 선택항목 z로부터 이름 붙은 완충기들은 제거되거나 잘리운 본문을 보관할때 사용자들이 사용할수 있다.

$ 현재행의 끝으로 이행한다. 앞에 놓인 수는 전진시켜야 할 행의 개수를 규정한다.

%　　현재유표위치에서 들어　맞추어 지는 ( )　혹은　{ }로 이동한다.

&　　ex의 &지령과 동등하다. 즉 앞선 교체지령을 반복한다.

'　　' 이 뒤에 오면 vi는 유표를 행의 시작위치에 놓으면서 이전의 지령상태로 귀환한다. 이전의 지령상태는 비상대적인 이행이 진행될 때마다 설정된다. a~z의 글자가 뒤에 오면 그 글자로 표식하여 놓은 행의 첫 비공백기호로 귀환한다.

d와 같은 연산자와 함께 리용될 때에 연산은 행전체에 작용한다.

`　　` 이 뒤에 올 때 vi는 유표를 행의 시작위치에 놓으면서 이전의 지령상태로 귀환한다. 이전의 문맥은 비상대적인 이행이 진행될 때마다 설정된다. a가　z의 글자가 뒤에 오면 그 글자로 표식해 놓은 행의 첫 비공백기호로 귀환한다.

d와 같은 연산자와 함께 리용될 때에 정확히 표식이 붙은 위치로부터 그 행안의 현재위치까지에서 작용한다.

[[　　이전의 구획경계에로 되돌아 간다. 구획은 해당한 선택항목의 값들로 정의된다. 새 행시작 ^L이나 { 에 의하여 시작된 행들도 [[에서 정지한다.

lisp에 대한 선택항목이 설정되면 유표는 행의 시작에 있는 매 (에서 정지한다.

^　　현재행의 첫 공백이 아닌 위치로 이행한다.

(　　문장의 시작으로 되돌아 간다. 문장은 ., !, ?뒤에 행의 끝기호 혹은 두개의 공백이 놓이면서 끝난다. 임의의 개수의 기호), [, ", ' 들이 ., !, ?와 공백 혹은 행의 끝사이에 놓일수 있다. 개수가 규정되면 유표는 규정된 개수의 문장만큼 후진방향에로 이동된다.

lisp에 대한 선택항목이 설정되면 유표는 lisp의 s-식의 시작에로 이행한다. 문장은 (혹은 구획경계에서 시작할수도 있다.

)　　문장의 처음위치로 전진방향에로 이동한다. 개수가 규정되면 유표는 규정된 개수의 문장만큼 전진방향에로 이동된다.

{　　앞선 단락의 시작으로 후진방향에로 이행한다. 단락은 선택항목 ( )의 값으로 정의된다. 완전히 빈 행과 구획경계도 단락의 시작으로 될수 있다. 개수가 규정되면 유표는 규정된 개수의 단락만큼 후진방향에로 이행된다.

}　　다음단락의 시작으로 전진방향에로 이행한다. 개수가 규정되면

유표는 규정된 개수의 단락만큼 전진방향에로 이행된다.

| 유표는 현재행의 규정된 렬로 이행한다. 앞의 수가 반드시 필요하다.

+ 다음행의 첫 비공백기호로 이행한다. 개수가 규정되면 유표는 규정된 개수의 행만큼 전진하는 방향으로 이동된다.

,(반점) 마지막 f, F, t, T지령들의 거꿀작용을 수행한다. 현재행에서 반대방향으로 탐색을 한다. 개수가 규정되면 유표는 규정된 회수만큼 탐색을 반복한다.

- 앞선 행의 첫 비공백기호로 이행한다. 개수가 규정되면 유표는 규정된 개수의 행만큼 후진방향으로 이동된다.

_(밑선) 현재행의 첫 비공백기호로 이행한다. 개수가 규정되면 유표는 규정된 개수의 행만큼 전진방향으로 이동된다. 이때 현재행은 첫행으로 된다.

.(점) 완충기를 변경시킨 마지막지령을 반복한다. 개수가 규정되면 규정된 회수만큼 그 지령을 반복한다.

/ 화면의 마지막행으로부터 기호렬을 읽고 그것을 정규식으로 해석하며 정합기호렬의 다음번 발생을 전진하는 방향으로 탐색한다. 탐색은 사용자가 패턴입력을 끝내는 <Enter>을 누를 때에 시작한다. 탐색은 SIGINT(혹은 사용자가 정의한 중단기호)를 보내는것으로서 끝낼수 있다.

본문의 확장을 규정하는 연산자가 함께 리용될 때에 정의된 령역은 현재 유표위치에서 시작하며 정합된 기호렬의 시작에서 끝난다. 전체행들은 정합된 행으로부터의 변위를 줌으로써 규정될 수 있다.

0 현재행의 첫기호에로 이동한다.

: ex지령을 시작한다. :과 입력된 지령은 제일 아래행에 현시된다. ex지령은 사용자가 <Enter>을 누를 때 실행된다.

; f, F, t, T를 리용하여 찾은 마지막 한개 기호를 반복한다. 만일 개수가 규정되면 탐색은 규정된 회수만큼 반복된다.

< 행들을 하나의 이행너비만큼 왼쪽으로 이행시키는 연산자이다. <뒤에는 행을 이행시키는 지령이 놓인다. 앞의 수는 이행지령에 통과된다.

<<는 현재행을 이행시킨다.

> 행들을 하나의 이행너비만큼 오른쪽으로 이행시키는 연산자이다.

=	선택항목 lisp가 설정되면 =는 규정된 행들이 lisp와 자동들여쓰기설정으로 입력된것처럼 재들여쓰기화를 한다. =앞에는 처리할 행개수를 지적하는 개수가 놓이며 뒤에는 이행지령이 놓인다.
?	/의 거꿀로서 후진하는 방향으로 탐색한다. /을 참고.
@buffer	buffer이름으로 기억된 지령들을 수행한다. <Enter>가 지령흐름의 어떤 부분이 아니라면 완충기내용의 끝에 <Enter>기호가 포함되지 말아야 한다. ex방식에서 수행되는 지령들은 :이 앞에 놓여야 한다.
~	유표에 의하여 매 기호를 구별하는 선택항목이다. 앞에 놓인 수는 몇개의 기호들이 현재의 행에서 구별되는가를 규정한다.
A	행의 끝에 본문을 추가하도록 한다. $a와 동등하다.
B	유표를 한 단어만큼 후진하는 방향으로 움직인다. 규정된 개수는 후진방향으로 옮기는 단어의 개수이다.
C	현재행의 나머지본문을 변경한다. c$와 동등하다.
D	현재행의 나머지본문을 제거한다. d$와 동등하다.
E	단어의 끝으로 전진하여 이동한다. 규정된 수는 전진하는 방향으로의 단어의 개수이다.
F	하나의 기호가 뒤에 놓여야 한다. 현재행에서 후진하면서 그 기호를 탐색하며 유표를 거기에 이행시킨다. 규정된 수는 탐색이 반복되는 회수이다.
G	앞의 인수로서 주어 지는 행번호로 이행하거나 앞의 인수가 없으면 파일의 끝으로 이행한다.
H	유표를 화면의 제일 웃행으로 이행한다. 수가 규정되면 그 수의 행개수만큼 화면의 제일 웃행으로부터 유표가 움직인다. 유표는 그 행의 첫 비공백기호에 놓인다.
I	행의 시작에서 삽입한다.
J	현재행을 다음행과 련결한다. 이때 단어들사이에는 하나의 공백, . 뒤에는 두개의 공백 그리고 다음행의 첫 기호가 닫는 괄호)이면 아무런 공백도 주지 않는다. 규정된 수는 그 수만큼의 행들이 련결되도록 한다.
L	화면의 마지막행의 첫 비공백기호로 유표를 이동시킨다. 규정된 수는 화면의 마지막부터 그 수만큼의 행을 이동시킨다. 연산자와 함께 리용되면 전체 행이 영향을 받는다.

M 유표를 화면의 중간행에 이동시키고 그 행의 첫 비공백위치에 놓는다.

N / 혹은 ?로 주어 진 마지막패턴의 다음번 정합을 진행한다. 그러나 n의 반대방향으로 진행된다.

O 현재행우에 있는 새행을 열고 입력방식을 입장시킨다.

P 유표의 앞에 마지막으로 제거된 본문을 다시 놓는다. 본문이 행 전체인 경우에는 유표의 우에 놓인다. 그밖에는 본문이 유표의 앞에 삽입된다.

XPG4인 경우에는 유표가 삽입된 기호들의 마지막렬위치에로 이동된다. P앞에 완충기지적자가 놓이면 그 완충기내용이 재탐색된다.

Q vi를 끝내고 ex지령방식을 입장시킨다.

R 화면의 기호들을 입력되는 기호들로 바꾼다. 이것은 입력이 escape건으로 끝날 때까지 계속된다.

S 전체 행들을 변경한다. cc와 동등하다. 앞의 수는 변경시키는 행개수를 규정한다.

T 반드시 하나의 기호가 뒤에 놓여야 한다. 그 기호를 현재행에서 후진하는 방향으로 탐색하며 유표를 그 기호의 바로 뒤에 놓는다. 규정된 수는 반복회수이다.

U 유표가 마지막으로 움직인 이전의 상태로 현재행을 재기억시킨다.

XPG4인 경우에 유표의 위치는 1번렬의 위치에 자동들여쓰기가 설정되였다면 이전의 행에서 지적한 위치에 설정된다.

W 현재행에서 단어의 시작으로 전진하는 방향으로 이행한다. 만일 현재위치가 단어의 시작이라면 현재 위치는 다음 시작단어의 첫 기호로 이동한다. 현재행에 부분적인 시작단어가 없으면 현재 위치는 시작단어를 포함하는 다음행의 첫 기호에로 이행한다. 이 지령에서 빈 행은 하나의 시작단어를 포함한것으로 고찰한다. 현재행은 선택된 시작단어를 포함하는 행으로 되며 현재위치는 선택된 시작단어의 첫 기호로 규정된다. 앞의 수는 전진하는 단어의 개수를 규정한다.

X 유표앞에 있는 기호를 제거한다. 규정된 수만큼 이 작용을 반복하지만 현재행에 있는 기호들만 제거한다.

Y 현재행을 복사하여 이름이 없는 완충기에 놓는다(yy와 동등하

다.). 규정된 수만큼의 행들이 완충기에 복사된다. Y앞에 완충기이름이 있으면 그 완충기에 행들을 복사한다.

ZZ 편집기에서 탈퇴한다. 그리고 마지막쓰기로부터 어떤 변화가 있었다면 완충기를 쓰면서 탈퇴한다.

a 입력방식을 입장시켜 입력된 본문을 현재 유표뒤에 추가한다. 규정된 수만큼의 본문이 삽입되게 된다. 그러나 삽입된 본문은 모두 한행에 있어야 한다.

b 현재행에서 앞단어의 시작으로 후진하는 방향으로 움직인다. 규정된 개수는 후진하는 방향으로 움직이는 단어의 개수이다.

c 이행지령이 반드시 뒤에 놓여야 한다. 주어 진 령역의 본문을 제거하고 입력방식을 입장시켜 제거된 본문과 새 본문을 교체한다. 한개 행이상이 영향을 받으면 제거된 본문은 수값완충기들에 보관된다. 현재행만이 영향을 받는다면 제거된 마지막 기호는 $로서 표식이 붙는다. 규정된 수만큼 이행지령이 반복된다. 만일 지령이 cc라면 전체 현재행이 변한다.

d 이행지령이 반드시 뒤에 놓여야 한다. 주어 진 령역의 본문을 제거한다. 한개 행이상이 영향을 받으면 본문은 수값완충기들에 보관된다. 규정된 수만큼 이행지령이 반복된다. 만일 지령이 dd 라면 전체 현재행이 제거된다.

e 다음 단어의 끝으로 전진하여 이동한다. 규정된 수만큼 반복하여 영향을 준다.

f 하나의 기호가 반드시 뒤에 놓여야 한다. 현재행의 나머지부분에서 그 기호를 탐색하며 유표를 거기에 이행시킨다. 규정된 수는 탐색이 반복되는 회수이다.

h 유표를 한 기호만큼 왼쪽으로 이행시킨다. ^H와 동등하다. 규정된 회수만큼 작용을 반복한다.

i 입력방식을 입장시키고 유표앞에 입력된 본문을 삽입한다(a를 참고).

j 유표를 동일한 렬에서 한행 아래로 이동시킨다. ^J, ^N와 동등하다.

k 유표를 동일한 렬에서 한행우로 이행시킨다. ^P와 동등하다.

l 유표를 한 기호만큼 오른쪽으로 이행시킨다. space와 동등하다.

nx 유표의 현재 위치에 표식을 붙인다. x는 a~z사이의 소기호로서 ' 및 `와 함께 리용되여 표식붙은 행을 참조하게 한다.

n　　지령들에서 마지막 /이나 ?를 반복한다.

o　　현재행아래에 한 행을 열고 입력방식을 입장시킨다. 그밖의 경우에는 O와 같다.

p　　유표의 뒤에 본문을 놓는다. 그밖의 경우는 P와 같다.

r　　하나의 기호가 반드시 뒤에 놓여야 한다. 그 기호는 유표아래에서 규정된 기호와 교체되여야 한다(새 기호는 새행이 될수 있다.). r앞의 수만큼 기호들을 규정된 기호로 교체한다.

s　　유표아래에 있는 하나의 기호를 제거하고 입력방식을 입장시킨다. 입력된 본문은 제거된 기호와 교체된다. 앞의 수는 현재행에서 몇개의 기호들이 변경되는가를 규정한다. 변경되는 마지막 기호에는 $로 표식이 붙는다.

t　　하나의 기호가 반드시 뒤에 놓여야 한다. 행의 나머지부분에서 그 기호를 찾는다. 유표는 찾아 진 기호의 앞렬에 이동한다. 앞의 수는 반복되는 탐색회수와 같다.

u　　현재완충기에 만들어 진 마지막변화를 취소한다. 다시 반복되면 이 두개의 상태가 서로 바뀐다. 한개행이상의 본문의 삽입후에 리용되였을 때 행들은 수자적완충기에 보관된다.

w　　다음단어의 시작으로 전진하는 방향으로 이동한다. 앞의 수는 유표가 전진하는 단어의 개수이다.

y　　반드시 이행지령의 뒤에 놓여야 한다. 규정된 본문은 이름이 없는 림시완충기에 복사된다. 이름이 있는 완충기규정이 앞에 있으면 본문은 그 완충기에도 놓인다. 만일 지령이 yy이면 전체 현재행이 선택된다.

z　　다음의 선택항목들에 의하여 현재행을 놓으면서 화면을 다시 그린다. z <Enter>는 화면의 꼭대기를 규정한다. z. 은 화면의 중심을, z-는 화면의 아래를 규정한다. z^, z+는 ^B, ^F와 비슷하다. 그러나 z^과 z+는 두개의 행이 겹치게 하지 않는다. z의 뒤 및 다음기호앞에 놓이는 수는 다시 그리는 화면에서 현시되는 행개수를 규정한다.

건반편집건들

초기화에서 편집기는 일부 말단건반편집건들을 동등한 시각적방식의 지령들에 자동적으로 대응시킨다. 이러한 대응은 아래의 표에서 주어 진 건들에서만 가능하며 terminfo(4)자료기지에서 정의된다.

지령 및 입력방식대응이 둘다 창조된다(ex(1)의 map지령을 참고). 입력방식을 단순히 절환(on/off)하는 삽입기호건들을 제외하고 입력방식대응은 입력방식을 끝내며 지령

방식대응으로서의 동일한 작용을 수행하고 그다음에 다시 입력방식을 입장시킨다.

일부 말단들에서 건반편집건들이 보내는 기호렬은 시각적방식지령에 대응된다. 또한 이런 기호렬은 사용자가 입장시켜 수행할수 있는 또 다른 지령 혹은 지령모임에 대응될수 있다.

이 기능은 입력방식대응에서 아주 효과적이다. 이러한 말단들에서 입력방식대응은 표준적으로는 불가능하다.

사용자들은 지령 및 입력방식의 건반편집건대응을 둘 다 선택항목 keyboardedit, keyboardedit!에 의하여 가능하게 하거나 불가능하게 하는것을 금지할수 있다(ex(1)을 참고).

선택항목 timeout, timeoutlen, doublespace들은 이 문제를 위한 여러가지 방법들을 준다.

terminfo 구체례	**지령방식 대응**	**입력방식 대응**	**대응이름**	**설 명**
key_ic	i	^[	inschar	기호삽입
key_eic	i	^[	inschar	기호를 끝에 삽입
key_up	k	^[ka	up	↑
key_down	j	^[ja	down	↓
key_left	h	^[ha	left	←
key_right	l	^[la	right	→
key_home	H	^[Ha	home	<Home>
key_il	o^[	^[o^[a	insline	행삽입
key_dl	dd	^[dda	delline	행제거
key_clear	^L	^[La	clear	화면청소
key_eol	d$	^[d$a	clreol	행청소
key_sf	^E	^[Ea	scrollf	아래로 scroll
key_dc	x	^[xa	delchar	기호제거
key_npage	^F	^[Fa	npage	다음페지
key_ppage	^B	^[Ba	ppage	이전페지
key_sr	^Y	^[Ya	sr	우로 scroll
key_eos	dG	^[dGa	clreos	화면의 끝까지 청소

외부적영향

환경변수들

UNIX 95는 이 지령에 대하여 XPG4작용을 리용하여 규정한다.

COLUNMS는 체계에서 규정된 화면의 수평크기를 무시한다.

LINES는 체계에서 규정된 화면의 수직크기를 무시하며 한개 화면에서는 행개수로서 리

용되고 시각적방식에서는 화면의 수직크기로서 리용된다.

SHELL은 !string형식의 연산인수와 함께 !, shell, read, 기타 지령들의 사용을 위한 지령행해석자로서 해석되는 변수이다. 쉘지령에서 프로그람은 -c와 기호렬 두개의 인수를 가지고 입장된다. 이 변수가 null이거나 설정되지 않으면 봉사프로그람 sh가 리용된다.

TERM은 말단의 종류이름으로서 해석되는 변수이다. 이 변수가 null이거나 설정되지 않았으면 기정의 말단종류가 리용된다.

PATH는 편집기지령들, shell, read, right에서 규정되는 쉘지령에 대한 탐색경로를 결정한다. exinit는 편집기를 시동할 때 첫 파일을 읽기전에 수행되는 ex지령들의 목록을 결정한다. 이 목록은 |기호로서 분리되는 여러개의 지령들을 포함할수 있다.

HOME은 편집기시동파일. exrc를 탐색하기 위한 등록부의 경로이름을 결정한다.

LC_ALL은 지역분류를 위한 임의의 값들을 무시하도록 하는 지역을 결정한다. 이 값들은 LANG 혹은 LC_으로 시작하는 환경변수의 설정에 의하여 규정된다.

LC_MESSAGES는 표준적인 오유장치에 씌여 지는 진단통보의 형식과 내용 그리고 표준적인 출력장치에 씌여 지는 비형식적인 통보에 영향을 미치는 지역을 결정한다.

LC_COLLATES는 정규식들을 평가하고 tag파일들을 처리하는데 리용되는 코드모임을 결정한다.

LC_CTYPE는 본문의 단일/다중바이트기호들, 대소기호의 분류, 대소기호들사이의 이행 그리고 정규식에 있는 기호클라스식들에 의하여 정합되는 기호들에 대한 해석을 결정한다.

LANG은 통보가 현시되는 언어를 결정한다.

LANGOPTS는 오른쪽으로부터 왼쪽으로 쓰는 언어에서 본문이 입력 및 출력파일에 기억되는 방법을 주는 선택항목들을 결정한다.

LC_COLLATES 혹은 LC_CTYPE가 환경에서 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG의 값이 기정값으로 리용된다. 만일 LANG이 규정되지 않았거나 빈 기호렬이면 "C"가 기정값으로 리용된다(lang(5)를 참고). 어떤 국제화변수가 틀린 설정을 포함하면 편집기는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

경고

ex(1)의 경고항목을 참고

프로그람제한

vi는 다음의 제한값들을 편집되는 파일에 놓는다.

최대행길이

LINE_MAX길이의 기호렬. 이것은 <limit. h>에서 정의되며 우의 2 ~ 3 바이트까지 포함된 값이다. 즉 LINE_MAX값이 2048일 때 2044개 까지의 기호렬이 가능하다.

이런 제한보다 긴 행을 포함하는 파일을 적재할 때 그 행들은 최대길이로 잘리운다. 그 파일을 보관하면 초기파일보다 잘리운 내용이 씌여지며 초기행들은 완전히 겹치게 된다.

편집기의 최대값보다 긴 행들을 창조하려고 하면 행이 너무 길다는 오유통보가 나온다.

최대파일크기

최대파일크기는 234,239행이다.

기타 제한들:

- 대역적인 매개 지령목록은 256개의 기호들까지 가능하다.
- vi 혹은 ex열기방식에서 128개 기호까지의 파일이름이 가능하다.
- 이전의 삽입/제거완충기의 크기는 128개 기호까지 가능하다.
- 쉘탈퇴지령은 100개 기호까지 가능하다.
- 기호렬값선택항목은 63개 기호까지 가능하다.
- 프로그람태그이름은 30개 기호까지 가능하다.
- map지령에 의하여 32개이하의 마크로를 정의할수 있다.
- 결합된 map마크로에서 총 기호수는 512이하이다.

저자

vi는 캘리포니아종합대학 버클리에 의하여 개발되였다. vi에 대한 16비트확장은 토시바회사의 쏘프트웨어에 토대하고 있다.

관련항목

ctags(1), ed(1), ex(1), stty(1), write(1), terminfo(4), envrion(5), lang(5), regexp(5)

The Ultimate Guide to the vi and ex Text Editors, Benjamin/Cummings Publishing Company, Inc. , ISBN 0-8053-4460-8, HP part number 97005-90015

표준일치

vi: SVID2, SVID3, XPG2, XPG3, XPG4

제9장. Bash쉘에 대한 개괄

각이한 쉘

대부분의 UNIX변종들에는 여러가지 쉘들이 있다. 쉘을 통하여 체계를 쉽게 리해할수 있기때문에 사용자들에게 있어서 쉘은 매우 중요하다. 쉘들에서 지령들을 주고 사용자환경들을 조종하며 지령파일들과 쉘프로그람들을 기입하는 등 UNIX변종들에서 쉘들을 리용하는 방법은 류사하다. 쉘에서는 대체로 많은 시간이 소비되기때문에 여러가지 각이한 쉘들을 리해하고 그것들중의 하나를 선택해야 한다. Bash 같은 특수한 쉘들은 UNIX변종들에서 그리 차이나지 않는다. 쉘들은 자체의 고유한 특징들을 가진다. 일반적으로 사용자들은 어느 한개 쉘을 특별히 좋아 하게 된다. 모든 쉘들은 기능적으로 류사하며 그것을 안 다음에는 리용하기가 재미 있다. 대부분의 UNIX변종들에서 체계관리자는 사용자들을 등록할 때 여러가지 각이한 쉘들중에서 하나를 선택할수 있으므로 사용자들이 실행시키는 쉘들을 유연하게 처리할수 있다. 일반적으로 체계관리자들은 체계관리를 쉽게 하기 위하여 사용자들이 동일한 쉘을 리용할것을 요구한다. 그러나 체계관리자들은 사용자가 체계에 있는 특정한 쉘을 리용해야 할 리유가 있으면 그것에 대한 사용자의 요구를 들어 주기도 한다. 이 장에서는 Bash쉘을 취급하고 다음의 두개 장들에서는 C쉘과 Korn쉘을 취급한다.

Bash쉘에 대한 개괄

대부분의 UNIX변종들에는 여러개의 쉘들이 있다. Linux에 기초한 UNIX조작체계들은 Bash쉘을 기정쉘로 리용한다. Bash쉘은 다른 쉘들의 좋은 특성만을 가지고 있으며 그 이름은 Bourne Again Shell에서 유도되였다. Bash쉘은 UNIX와의 사용자대면부를 제공한다는 면에서 다른 쉘들과 류사하다. Bash쉘을 다음과 같은 3가지 방식으로 리용할수 있다.

- 대화적으로 지령행에 지령들을 입력한다.
- 흔히 리용되는 지령모임들을 지령파일에 그룹화하고 그 파일의 이름을 입력하여 실행시킨다.
- 쉘의 구조화된 프로그람작성기술을 리용하여 Bash쉘프로그람들을 작성한다.

이 3가지 기술들은 자주 리용되는 순서로 렬거되였다. 우선 체계에 가입하고 대화적 지령들을 리용한다. 그다음 흔히 리용되는 지령들을 그룹화하고 그것을 하나의 지령으로 실행시킨다. 끝으로 쉘스크립트들을 창조한다.

그러므로 여기서는 Bash쉘의 개념들을 우에서 렬거된 순서로 서술한다. Bash쉘의 지령파일들과 프로그람작성개념들은 《쉘프로그람작성》장에서 취급된다. Bash쉘은 《쉘프

로그람작성》장에서 리용되는 쉘인 Korn쉘과 매우 류사하다. 그러므로 《쉘프로그람작성》장에서 취급되는 방법들을 Bash프로그람작성에서도 리용할수 있으나 실지 프로그람을 작성할 때에는 항상 차이가 있다는것을 명심하여야 한다.

지령주기

체계에 가입한 다음 사용자가 수행하는 첫 작업은 **입력재촉문**(Prompt)에 따라 지령을 주는것이다. 처음으로 주어야 할 지령은 ls -al이다. 현작업등록부가 뿌리등록부일 때 이 지령을 주면 다음과 같은 파일들이 렬거된다.

```
# pwd
# le -al
total 46
drwxr-xr-x   5  root  root   1024  Nov  26  19:40  .
drwxr-xr-x  20  root  root   1024  Nov   8  20:10  . .
-rw-r--r--   1  root  root    964  Nov  26  19:40  . bash_history
-rw-r--r--   1  root  root    674  Feb   5   1997  . bashrc
-rw-r--r--   1  root  root    602  Feb   5   1997  . cshrc
-rw-r--r--   1  root  root  14815  Nov   8  20:09  . fvwmrc. menus. prep
-rw-r--r--   1  root  root    116  Feb   5   1997  . login
-rw-r--r--   1  root  root    234  Feb   5   1997  . profile
drwxr-xr-x   2  root  root   1024  Nov   8  14:10  . seyon
-rw-r--r--   1  root  root   4276  Nov   8  20:09  XF86Config
-r--r--r--   1  root  root  13875  Nov   8  20:05  XF86Config. bak
drwxrwxrwx   2  root  root   1024  Nov  26  19:40  book
drwxr-xr-x   5  root  root   1024  Nov  14  18:12  lg
-rw-r--r--   1  root  root      0  Nov  26  19:40  typescript
```

이 파일들중에는 .bashrc라고 부르는 Bash시동파일이 있다. 아래에 .bashrc파일의 내용을 보여 준다.

```
# cat .bashrc
# ~/. bashrc - -
# The individual per-interactive-shell startup file for bash
. /etc/profile
# try solve this tedious 'Backspace vs. Delete'  problem . . .
if [ -z "$TERM" ]; then
    echo ". bashrc: TERM empty: this shouldn't happen!" 1>&2
```

```
echo " Please contact ' support@lst.de' "1>&2
else
    case $TERM in
    linux*)
     stty erase '^?'
     ; ;
    *)
     stty erase ' ^ H '
     ; ;
    esac
fi

# general environment settings
# export GROFF_TYPESETTER=latinl
# export LC_CTYPE=iso-8859-1
    export LESSCHARSET=latinl
#export METAMAIL_PAGER=less

HISTSIZE=100

alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend
#
```

.bashrc파일에는 HISTSIZE에 대한 값과 별명모임이 있다. 이 별명들은 긴 지령들에 대한 간략지령이다. 실례로 지령 ll을 주면 실지로는 지령 ls -Alg을 준것과 같다. 우에 렬거된 파일목록에서 보여 주는바와 같이 국부적인 .profile뿐아니라 /etc/profile도 실행시킬수 있다. /etc/profile은 보통 체계에 가입한 모든 사용자들을 등록한다. 아래에 /etc/profile의 내용을 보여 준다.

```
# cat /etc/profile
# /etc/profile
# System wide environment and startup programs
# Functions and aliases go in $HOME/.bashrc
```

```
PATH="/bin:/usr/bin:/opt/bin:/usr/XllR6/bin:/usr/openwin/bin:/usr/TeX/bin:/usr/
local/bin"

umask 022

if [ `id -gn` = `id -un`] && [ `id -u` !=0] ; then
 umask 002
fi

if [ -z "$UID" ] ; then
 UID=`id -u`
fi

if [ "$UID" = 0] ; then
    PATH=/sbin:/usr/sbin:$PATH
else
    PATH=$PATH:
fi
USER=`id -un`
LOGNAME=$USER

export PATH USER LOGNAME

HOSTNAME=`/bin/hostname`
MAIL="/var/spool/mail/$USER"

export HOSTNAME MAIL

if [ -n "$BASH_VERSION" ]; then
 #（aliases now in $HOME/.bashrc，resp．/etc/skel/.bashrc）
 export PS1="[\u@\h \W]\\$"
 export HISTSIZE=100
fi
#
```

여기서는 /etc/profile의 일부 내용들도 취급한다.

.bashrc안의 리력목록 초기화

Bash쉘은 사용자가 실행시킨 지령들의 **리력목록**(history list)을 가진다. 현 지령을 다시 주거나 이미 준 지령을 보려면 리력목록을 리용할수 있다.

또한 리력목록에 포함될 지령들의 수를 규정할수 있다. 다음의 행은 .bashrc에서 리력목록을 100으로 설정한다.

```
set history=100
```

그러면 작업을 끝낼 때 사용자가 준 마지막 100개의 지령들이 리력목록에 보관된다. 다시 체계에 가입하면 이 100개의 지령들을 볼수 있다. 새로운 지령들을 주면 리력목록에서 가장 오랜 지령들은 제거된다. 이 사실을 다음의 실례가 보여 준다.

```
# history
  2     more history
  3     ll
  4     cd . .
  5     pwd
  6     cd . .
  7     ll
  8     cd . .
  9     ll
  10    ll log
  11    cd log
  12    more *
  13    l
  14    ll
  15    cd /
  16    ll
  17    cd
  18    XF86Setup
  19    XF86Setup
  20    startx
  21    ll
  22    pwd
  23    ll
  24    ll /
  25    XF86Setup
  26    ll
  27    startx
  28    find / -name XF86Config*
  29    cp /usr/XllR6/lib/Xll/XF86Config.eg .
  30    ll
  31    XF86Setup
  32    XF86Setup
  33    startx
```

```
34    ll
35    mv XF86Config.eg XF86Config
36    XF86Setup
37    startx
38    shutdown -h now
39    man ls
40    man ll
41    man ls
42    man file
43    lsr
44    man chmod
45    man chmod
46    shutdown -h now
47    pwd
48    ls -l
49    pwd
50    ls -a
51    ls -al
52    pwd
53    ls -al
54    more    .profile
55    more    .bashrc
56
57    alias
58    ll
59    pwd
60    script
61    script
62    scrit
63    script
64    more .bashrc
65    more .bashrc
66    ll
67    more .profile
68    ll
69    more .bashrc | grep P
70    more .profile | grep P
71    env
72    more /.profile
```

```
73      more /etc/profile
74      more /etc/profile | grep PS
75      find / -name *profile* -print
76      more .bashrc
77      more .bashrc
78      ll /etc/profi*
79      cp /etc/profile /etc/profile.orig
80      vi /etc/profile
81      exit
82      cp /etc/profile.orig /etc/profile
83      history
84
85      exit
86      history
87      history
88      ll
89      ll
90      history
91      ll /etc/profi*
92      ll /etc/profi*
93      more .bashrc
94      history
95      ll
96      cd /root
97      ll
98      history | more
99      history | more
100     exit
101     history
```

이 실례에서 지령번호 100은 exit 또는 작업을 끝내는 지령이다. 지령번호 101은 다음작업을 시작하기 위해 주는 history지령이다.

리력목록으로부터의 재호출

이 모든 지령들(cp, more, find, ll)은 리력목록에서 대응하는 번호를 가진다. 마지막지령을 **!!**으로, 89번째 지령을 !89로, "m"으로 시작되는 마지막지령을 !m으로 반복할 수 있다. 이것을 아래의 실례에서 보여 주고 있다.

```
# !!
history
    3     ll
    4     cd ..
    5     pwd
    6     cd ..
    7     ll
    8     cd ..
    9     ll
    10    ll log
    11    cd log
    12    more *
    13    l
    14    ll
    15    cd /
    16    ll
    17    cd
    18    XF86Setup
    19    XF86Setup
    20    startx
    21    ll
    22    pwd
    23    ll
    24    ll /
    25    XF86Setup
    26    ll
    27    startx
    28    find / -name XF86Config*
    29    cp /usr/XllR6/lib/Xll/XF86Config.eg .
    30    ll
    31    XF86Setup
    32    XF86Setup
    33    startx
    34    ll
    35    mv XF86Config.eg XF86Config
    36    XF86Setup
    37    startx
    38    shutdown -h now
    39    man ls
```

```
40    man ll
41    man ls
42    man file
43    lsr
44    man chmod
45    man chmod
46    shutdown -h now
47    pwd
48    ls -1
49    pwd
50    ls -a
51    ls -al
52    pwd
53    ls -al
54    more .profile
55    more .bashrc
56
57    alias
58    ll
59    pwd
60    script
61    script
62    scrit
63    script
64    more .bashrc
65    more .bashrc
66    ll
67    more .profile
68    ll
69    more .bashrc | grep P
70    more .profile | grep P
71    env
72    more /.profile
73    more /etc/profile
74    more /etc/profile | grep PS
75    find / -name *profile* -print
76    more .bashrc
77    more .bashrc
78    ll /etc/profi*
```

```
    79      cp /etc/profile /etc/profile.orig
    80      vi /etc/profile
    81      exit
    82      cp /etc/profile.orig /etc/profile
    83      history
    84
    85      exit
    86      history
    87      history
    88      ll
    89      ll
    90      history
    91      ll /etc/profi*
    92      ll /etc/profi*
    93      more .bashrc
    94      history
    95      ll
    96      cd /root
    97      ll
    98      history | more
    99      history | more
    100     exit
    101     history
    102     history
# !89
    ll
total 44
-rw-r--r--   1  root  root    956  Nov  26  19:33  .bash history
-rw-r--r--   1  root  root    674  Feb   5   1997  .bashrc
-rw-r--r--   1  root  root    602  Feb   5   1997  .cshrc
-rw-r--r--   1  root  root  14815  Nov   8  20:09  .fvwmrc.menus.prep
-rw-r--r--   1  root  root    116  Feb   5   1997  .login
-rw-r--r--   1  root  root    234  Feb   5   1997  .profile
drwxr-xr-x   2  root  root   1024  Nov   8  14:10  .seyon
-rw-r--r--   1  root  root   4276  Nov   8  20:09  XF86Config
-r--r--r--   1  root  root  13875  Nov   8  20:05  XF86Config.bak
drwxrwxrwx   2  root  root   1024  Nov  13  21:25  book
drwxr-xr-x   5  root  root   1024  Nov  14  18:12  lg
-rw-r--r--   1  root  root      0  Nov  26  19:36  typescript
```

```
# !m
more .bashrc
# ~/.bashrc --
# The individual per-interactive-shell startup file for bash

. /etc/profile
#  try solve this tedious 'Backspace vs. Delete' problem...
if [ -z "VERM" ] ; then
   echo ".bashrc: TERM empty: this shouldn't happen!" 1>&2
   echo    Please contact 'support@lst.de'" 1>&2
else
   case $TERM in
   linux*)
    stty erase ' ^? '
    ; ;
 *)
    stty erase ' ^H '
   ; ;
   esac
fi

# general environment settings
# export GROFF-TYPESETTER=latinl
#export LC_CTYPE=iso-8859-1
[7m--More--(70%)[m
export LESSCHARSET=latinl
# export METAMAIL_PAGER=less
HISTSIZE=100
alias which='type -path'
alias h=history
alias j="jobs -l"
alias l="ls -Fax"
alias ll="ls -Alg"
alias pd=pushd
alias z=suspend

[root@nycald1 /root]#
Script done on Thu Nov 26 19:39:51 1998
```

표 9-1은 흔히 리용되는 리력목록재호출지령들을 보여 준다.

표 9-1 리력목록으로부터의 재호출

지 령	서 술	실 례
!*N*	지령 **N**을 실행	**!2**
!!	마지막지령을 실행	**!!**
!-*N*	마지막으로 실행된 지령으로부터 **N**번째 지령을 실행	**!-N**
!*str*	str로 시작되는 마지막지령을 실행	**!c**
!?*str*?	지령행의 그 어디에든지 str가 들어 있는 마지막지령을 실행	**!?cat?**
!{*str1*}*str2*	str1이 있는 마지막지령에 str2를 첨가	**!{cd}/tmp**
^*str1*^*str2*^	마지막지령에서 str1을 str2로 치환	**^cat^more^**

지령행의 편집

지령들을 보면서 다시 주는 좋은 방법은 리력목록을 리용하는것이다. Bash쉘에서는 **지령행**(Command Line)편집도 할수 있다. 우로의 화살건을 리용하여 리력목록에서 뒤로 한개 지령씩 이동할수 있다. 우로의 화살건을 누를 때 리력목록의 마지막지령은 지령행에 나타난다. 지령이 지령행에 나타날 때 <Enter>건을 누르면 그 지령이 실행된다. 왼쪽 또는 오른쪽 화살건을 리용하여 지령행의 필요한 위치에 옮겨 간 다음 보충적인 정보를 입력하거나 공백건 또는 삭제건으로 지령행의 정보를 제거하여 지령을 수정할수 있다.

.bashrc에 있는 별명

별명(Alias)은 자주 리용되는 지령 또는 지령렬에 붙여 주는 이름이다. 별명들은 파일 .bashrc에 보관되며 매번 체계에 가입할 때마다 이 파일을 읽을수 있게 한다. 앞에서 본 파일 .bashrc에는 이미 7개의 별명들이 들어 있다. 파일 .bashrc에 별명들을 추가하거나 지령행에서 별명들을 정의할수 있다. 그러나 이 별명들은 작업을 끝낼 때 지워 진다. 아래에서 .bashrc파일에 설정되여 있는 별명들의 목록과 별명 l과 ll을 실행시키는 실례를 보여 준다.

```
# alias
alias  h='history'
alias  j='jobs  -l'
alias  1='ls  -Fax'
alias  ll='ls  -Alg'
alias  pd='pushd'
```

```
alias which='type -path'
alias z='suspend'
#
# l
./          ../                 .bash_history       .bashrc
.cshrc      .fvwmrc.menus.prep  .login              .profile
.seyon/     XF86Config          XF86Config.bak      book/
lg/         typescript
#
# ll
total 44
-rw-r--r--   1 root  root    970  Nov 26  21:35 .bash_history
-rw-r--r--   1 root  root    674  Feb  5   1997 .bashrc
-rw-r--r--   1 root  root    602  Feb  5   1997 .cshrc
-rw-r--r--   1 root  root  14815  Nov  8  20:09 .fvwmrc.menus.prep
-rw-r--r--   1 root  root    116  Feb  5   1997 .login
-rw-r--r--   1 root  root    234  Feb  5   1997 .profile
drwxr-xr-x   2 root  root   1024  Nov  8  14:10 .seyon
-rw-r--r--   1 root  root   4276  Nov  8  20:09 XF86Config
-r--r--r--   1 root  root  13875  Nov  8  20:05 XF86Config.bak
drwxrwxrwx   2 root  root   1024  Nov 26  19:41 book
drwxr-xr-x   5 root  root   1024  Nov 14  18:12 lg
-rw-r--r--   1 root  root      0  Nov 26  21:35 typescript
#
```

이 별명들을 모두 그대로 쓸수 있지만 우리자체로 별명을 설정해 보자. 몇개의 프로쎄스들이 체계상에서 실행되고 있는가를 알려고 한다고 하자. 이것을 위해 "procs"라고 부르는 별명을 만든다. 지령 ps는 프로쎄스들을 렬거한다. ps와 파이프(|)를 주면 행의 개수를 선택항목 "l"이 있는 wc에 출력한다. 막대기는 ps의 출력이 wc의 입력으로 리용된다는것을 의미한다. ps는 프로쎄스들의 목록을 생성하고 wc -l은 행들의 개수를 주므로 실행되고 있는 프로쎄스들의 총수를 알게 된다. 다음의 실례는 우선 ps의 출력을 보여주고 그 다음에 지령 alias와 그 지령에 의하여 나타나는 출력을 보여 준다.

```
#  ps
  PID TTY  STAT   TIME COMMAND
  188   2    S    0:00 /sbin/getty tty2 VC linux
  189   3    S    0:00 /sbin/getty tty3 VC linux
  190   4    S    0:00 /sbin/getty tty4 VC linux
  191   5    S    0:00 /sbin/getty tty5 VC linux
  192   6    S    0:00 /sbin/getty tty6 VC linux
```

```
  619      1      S      0:00  login  root
  620      1      S      0:00  -bash
  642      1      S      0:00  script
  643      1      S      0:00  script
  644      p0     S      0:00  bash  -i
  656      p0     R      0:00  ps
#
# alias  procs='echo  "Number  of  processes  are:  ";ps  |  wc  -1'
#
#   procs
Number  of  processes  are:  ll
#
```

이 별명을 정의함으로써 체계상에서 실행되고 있는 프로쎄스들의 수를 알려고 하는 경우에 "procs" 만을 입력하면 된다. 이 지령행에는 많은 인용문들이 있다. 그 인용문들이 무엇인가를 알려면 표 9-2를 보면 된다.

표 9-2 쉘 인용문

기호들	설 명
'cmd'	웃반점은 기호렬이 문자그대로 취급된다는것을 의미한다.
"str"	웃두점은 지령 및 변수치환을 허락한다는것을 의미한다.
\c	esc기호인데 인쇄할 때 그뒤의 기호렬이 행바꾸기없이 인쇄되도록 한다.
'str'	이것은 지령을 실행하고 출력을 치환한다는것을 의미한다.

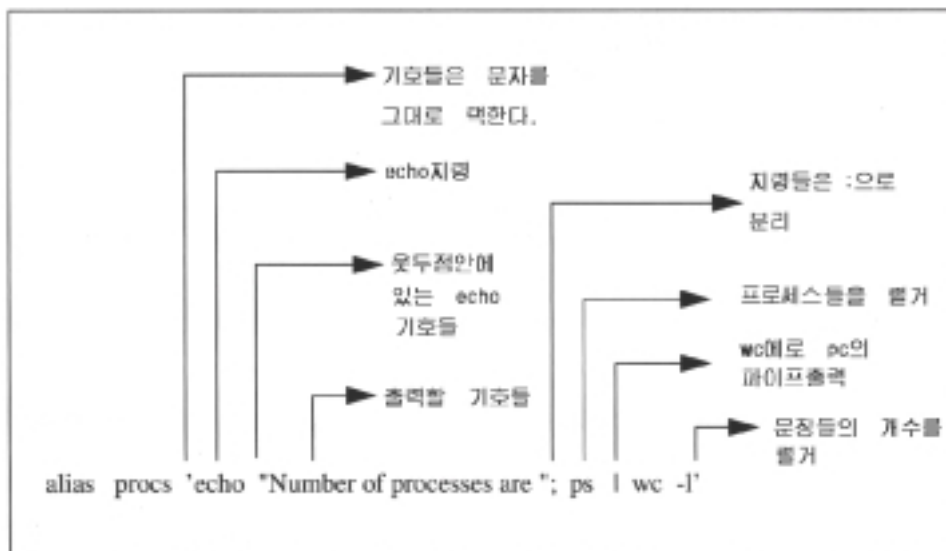



그림 9-1. 인용문실례

표 9-2를 별명 procs에 적용하면 이 별명이 무엇을 포함하는가를 알수 있다. 그 별명은 웃반점으로 시작된다. 웃반점안에 들어 있는것은 지령을 의미한다. 첫 지령은 지령 echo인데 이 지령은 출력할 기호렬을 웃두점안에 넣어 리용한다. 인쇄될 때 행바꾸기를 금지시키는 기호 \c를 첨가할수 있다. 두반점 (;)은 지령들을 분리시킨다. 그림 9-1에서 보여 주는바와 같이 ps가 실행되면 프로쎄스들의 목록이 wc에로 출력되여 프로쎄스들의 개수가 얻어 진다.

그림 9-1에서 볼수 있는바와 같이 일부 인용문들은 리해하기가 어렵다. 쉘스크립트들을 변경하거나 다시 리용하려면 인용문을 리해하는것이 중요하다.

지령과 경로의 완성

Bash쉘은 때때로 사람이 무엇을 생각하고 있는가를 알아 차린다. 지령 또는 경로이름의 일부분만을 입력하고 "tab"건을 누르면 그 지령이 완성된다. 실례로 현 체계의 실행준위를 보기 위하여 지령 runlevel을 주려고 할 때 그 지령이름을 잘 모른다고 하면 "run"을 입력하고 tab건을 누른다. 그러면 다음의 실례에서 보여 주는것처럼 지령이 완성된다.

run<tab key>**level**
N 3

Bash쉘은 "run"으로 시작되는 지령이 runlevel뿐이라는것을 확정하고 그 지령을 완성한다. 만일 지령 또는 경로이름이 유일하지 않으면 Bash쉘은 그 지령을 완성하기 위한 선택항목을 보여 준다. 아래에 "ru"를 입력하고 tab건을 두번 눌러 "ru"로 시작되는 지령들을 찾은 실례를 보여 준다.

ru<tab key><tab key>
runlevel rusers

이 실례로부터 "ru"를 입력하면 두개의 지령 runlevel과 rusers이 나타나는것을 볼수 있다. 경로이름에 대하여서도 마찬가지이다.
등록부를 "/b"로 교체하면 다음과 같은 결과를 얻는다.

cd /b<tab key><tab key>
bin boot

뿌리준위에 "b"로 시작되는 두개의 등록부가 있기때문에 Bash는 그것들중의 어느것을 의미하는지 결정할수 없었으며 따라서 그 두개를 모두 렬거하였다.

파일이름확장

파일을 조작하려면 사용자는 파일이름을 취급하는 쉘스크립트를 작성하여야 한다. 쉘스크립트를 작성하기전에 파일이름확장에 대하여 아는것이 좋다. 표 9-3은 몇가지 공통적인 파일이름확장과 패턴정합을 보여 준다.

표 9-3 파일이름확장과 패턴정합

기 호 들	실 례	설 명
*	1) ls*.c	령 또는 그 이상기호들을 맞춘다.
?	2) ls conf.?	임의의 한개 기호를 맞춘다.
[list]	3) ls conf.[co]	목록에 있는 임의의 기호를 맞춘다.
[lower-upper]	4) ls libdd.9873[5-6].sl	범위에 있는 임의의 기호를 맞춘다.
str{str1, str2, str3, …}	5) ls ux*.{700, 300}	str를 { }의 내용으로 확장한다.
~	6) ls -a~	홈등록부
~username	7) ls -a ~gene	username의 홈등록부

표 9-3에 있는 실례들을 자세히 서술하면 다음과 같다.

ㄱ) 등록부에서 ".c"로 끝나는 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls *.c
  conf.SAM.c conf.c
```

ㄴ) 등록부에서 "conf"로 이름지어 져 있고 한개 기호로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls conf.?
  conf.c   conf.o  conf.1
```

ㄷ) 등록부에서 "conf"로 이름지어 져 있고 한개 기호 "c" 또는 "o"로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls conf.{co}
  conf.c  conf.o
```

ㄹ) 류사한 이름들을 가지면서 어떤 범위의 마당들을 가지는 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls libdd9873[5-6].sl
  libdd98735.sl  libdd98736.sl
```

ㅁ) "ux"로 시작되고 확장자가 "300" 또는 "700"인 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls ux*.{700, 300}
  uxbootlf.700  uxinstfs.300
```

ㅂ) 홈등록부에 있는 파일들을 렬거하려면 아래에서 보여 주는것처럼 "～"를 리용

한다.

```
$ ls - a~
.       .cshrc.org      .login  .shrc.org
..      .exrc           .login.org      .cshrc
.history
```

ㅅ) 사용자의 홈등록부에 있는 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls -a ~gene
.              .history       splinedat      under.des
. .            .login         trail.txt      xtra.part
.chsrc         .login.org     ESP-File
.cshrc.org     .profile       Mail
.exrc .shrc.org      opt
```

파일이름확장에 익숙하여야 쉘스크립트를 쉽게 작성할수 있다.

방향바꾸기(I/O방향바꾸기)

UNIX는 지령들이 입력을 건반(표준입구장치)으로부터 받으며 출력을 화면(표준출구장치)에 보내도록 설치되여 있다. 지령들은 오유정보도 화면에 보낸다. 이 표준설정을 무시하고 입력이 다른 곳으로부터 들어 오고 출력과 오유들이 다른 곳으로 나가게 할수도 있다. 이것을 **방향바꾸기**(Redirection)라고 부른다. 표 9-4는 방향바꾸기의 형식들을 보여 준다. 표에서 보는바와 같이 지령의 출력을 표준출구장치로부터 파일에로 전환하려면 ">"을 리용하여야 한다. noclobber라고 부르는 환경변수를 설정하면 이미 존재하는 파일에로의 방향바꾸기는 진행되지 않는다. 실례로 다음의 /tmp/processes와 같은 이미 존재하는 파일에 기입하려고 시도한다면 그 파일이 존재한다는 통보문을 받는다.

```
# ps ?ef > /tmp/processes
/tmp/processes: File exists
```

그러나 파일을 덧 쓰도록 방향바꾸기에 "!"를 리용할수 있다. ">!"를 리용하면 파일이 덧 씌여 진다. 그리고 “>>!”은 이미 존재하는 파일의 뒤에 출력이 첨가되도록 한다. 이것에 대한 실례들을 표 9-4에서 보여 준다.

표 9-4　　　흔히 리용되는 방향바꾸기형식들

지령 또는 대입	실 례	설 명
<	wc -l < .login	표준입력방향전환 : wc(단어계수)를 실행하고 .login에 있는 행들의 수를 렬거한다.
>	ps -ef > /tmp/processes	표준출력방향전환 : ps를 실행하고 출력을 파일/tmp/process에로 보낸다.

(표계속)

지령 또는 대입	실 례	설 명
>>	ps -ef >> /tmp/processes	표준출력을 첨가 : ps를 실행하고 출력을 파일/tmp/process에 첨가한다.
>!	ps -ef > ! /tmp/processes	출력방향전환을 첨가하고 noclobber를 무시한다. /tmp/processes가 존재하면 그우에 덧쓴다.
>>!	ps -ef >>! /tmp/processes	표준출력을 첨가하고 noclobber를 무시한다. /tmp/processes의 끝에 첨가한다.
| (pipe)	ps | wc -l	ps를 실행하고 그 결과를 wc의 입력으로 리용한다.
0 - standard input		
1 - standard output		
2 - standard error	cat program 2> errors	파일program을 표준출력에 cat하고 오유들을 파일 errors에로 방향전환한다.
	cat program 2>> errors	파일program을 표준출력에 cat하고 오유들을 파일 errors에로 첨가한다.
	find / -name '*.c' -print >cprograms 2>errors	체계상에서 .c로 끝나는 한 파일들을 찾고 파일들의 목록을 현작업등록부에 있는 cprograms에 배치한다. 그리고 모든 오유들을 (파일서술자2) 현재작업등록부에 있는 파일 errors에 보낸다.
	flnd / -name '*.c' -print >cprograms 2>&1	체계상에서 .c로 끝나는 모든 파일들을 찾고 파일목록을 현재작업등록부에 있는 cprograms에 배치한다. 그리고 모든 오유(파일서술자2)들을 파일서술자1(cprograms)이 있는 동일한 위치에로 보낸다.

환경변수

환경변수(Environment Variable)는 기호렬에 결합된 이름이다. 이름은 변수이고 기호렬은 값이다. 체계상에서 지령을 줄 때 보통 절대적인 경로이름이 아니라 상대적인 경로이름을 입력한다. 변수 PATH는 지령들이 배치되여 있는 등록부의 위치를 가리킨다. 이 변수가 없으면 매 지령의 절대적인 경로이름을 지적해 주어야 한다. 지령을 줄 때 쉘은 변수 PATH에 렬거된 등록부들을 렬거된 순서대로 탐색한다. 설정된 환경변수들을 보기 위한 좋은 방법은 아래에서 보여 주는바와 같이 env지령을 리용하는것이다.

```
# env
HISTSIZE=100
HOSTNAME=nycaldl.hp.com
LOGNAME=root
MAIL=/var/spool/mail/root
TERM=linux
HOSTTYPE=i386
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/opt/bin:/usr/X11R6/bin:
/usr/openwin/bin:/usr/TeX/bin:/usr/local/bin
HOME=/root
SHELL=/bin/bash
PS1=[\u@\h   \W] \ $
USER=root
LESSCHARSET=latinl
OSTYPE=Linux
SHLVL=2
_=/usr/bin/env
#
```

우의 실례에서 보여 주는바와 같이 변수 PATH외에도 UNIX에서의 작업을 쉽게 해주는 다른 환경변수들이 있다. 특정한 환경변수의 값을 알려면 환경변수 HOME에 대한 다음의 실례에서 보여 주는바와 같이 지령 **echo를** 리용한다.

```
$ echo $HOME
/root
```

환경변수의 앞에 붙은 "$"은 그 변수의 값이 표준출구장치에로 나가야 한다는것을 규정한다. 이 경우에 환경변수 HOME의 값은 /root이다. 이것은 사용자가 /root의 홈등록부를 가진다는것을 의미한다.

Bash쉘에서는 다음과 같은 형식으로 자체의 환경변수를 정의한다.

```
export NAME=value
```

환경변수 PSI에 의하여 정의된 입력재촉문을 다음의 지령으로 설정할수 있다.

PS1=[\u@\h \w] \$

여기서 PS1는 여는 각괄호, 사용자이름, "at"표식(@), 호스트이름, 공백, 홈등록부, 닫는 각괄호, $로 설정되여 있다. 앞의 환경변수에서 볼수 있는바와 같이 PSI는 가장 복잡한 형식이다.

또한 이미 존재하는 변수의 끝에 다음과 같은 형식으로 첨가할수 있다.

export NAME="$NAME:appended_information"

/root/programs를 이미 존재하는 환경변수 PATH에 첨가하려면 다음과 같은 지령을 주어야 한다.

export PATH="$PATH: /root/programs "

이것은 경로 /root/programs 를 환경변수 PATH에 첨가한다.

환경변수들과 그것의 값들을 보려면 지령 env를 리용한다.

배경일감과 일감조종

지금까지 본 많은 실례들에서처럼 지령을 실행시키면 그 지령이 완성될 때까지 입력재촉문이 나타나지 않는다. 일부 지령들은 완성되는데 많은 시간을 소비하므로 입력재촉문이 다시 나타날 때까지 오래동안 기다려야 한다.입력재촉문이 다시 나타나기를 기다리면서 그 지령을 배경에서 실행시킬수 있다. UNIX는 다중과제체계이기때문에 많은 지령들을 배경에서 실행시키고 다른 지령들을 더 줄수 있도록 입력재촉문을 제공해 준다.

지령을 배경에서 실행시키려면 간단히 지령행의 끝에 &를 첨가하여야 한다. 지령의 끝에 한개의 &를 붙이면 그 지령은 배경에서 실행되며 입력재촉문은 곧 다시 나타난다. 체계상에 수많은 파일들과 등록부들이 있기때문에 모든 지령들을 렬거하고 렬거된 그 지령들을 파일안에 넣으려면 오랜 시간이 걸린다. 체계상에 얼마나 많은 파일들과 등록부들이 있는가를 보기 위한 지령을 실행시켜 보자. 다음의 실례는 ls지령을 선택항목 "a", "-l", "-R"를 붙여 실행시켜 체계상의 모든 파일들을 렬거하고 그 결과를 wc에로 보내여 행의 총 개수를 계산한것을 보여 준다.

```
# ls -alR | wc -l
ls: proc/18/exe: No such file or directory
ls: proc/19/exe: No such file or directory
ls: proc/2/exe:  No such file or directory
   29982
```

이 출력은 3개의 오유와 행의 총 개수가 29982이라는것을 보여 준다. 지령 ls -alR로 29982개 항목들을 모두 한개 파일에 기입하려면 입력재촉문이 나타날 때까지 앉아 기다려야 한다. 그러므로 모든 항목들을 파일에 기입하려는 경우에 지령의 끝에 &를 붙여

주는것이 좋다. 그러면 다음의 실례에서 보여 주는바와 같이 **일감**(Job)은 배경에서 실행되며 입력재촉문은 곧 되돌려 진다.

```
# ls -alR > /tmp/files &
[1] 817
ls: proc/18/exe: No such file or directory
ls: proc/19/exe: No such file or directory
ls: proc/2/exe:  No such file or directory
```

이 지령이 배경에서 실행된 결과는 각괄호안에 포함된 일감번호와 프로쎄스id 또는 PID이다. 이 지령을 수행하는 프로쎄스가 발생하는 오유들은 화면에 인쇄된다. 파일목록과 오유들이 /tmp/files에 기입되도록 하고 일감을 배경에서 실행시키려면 다음의 지령을 주면 된다.

```
# ls ?alR >& /tmp/files &
[1] 817
```

이렇게 하면 오유들이 파일목록과 함께 /tmp/files에 기입되므로 화면에서는 보이지 않으며 지령입력재촉문은 곧 다시 나타난다. /tmp/files를 들여다 보면 앞에서 발생된 3개의 오유들이 파일안에 들어 있는것을 볼수 있다.

배경에서 실행되지 않는 **전경일감**(Foreground Job)들과 **배경일감**(Background Job)들을 조종할수 있다. 전경일감을 정지시키려면 다음의 실례에서 보여 주는바와 같이 ctrl-z를 입력하면 된다.

```
# ls -alR
            .
            . .
            . lgb
            amd
            auto
            bin
ctrl - z
[4]+       Stopped ls -alR
```

ctrl-z를 누르면 지령 ls가 정지되고 일감번호 4와 그것의 상태 "Stopped"가 표시된다. 이 지령은 완전히 끝난것이 아니라 일시 정지되였을뿐이다. 이 프로쎄스를 fg지령으로 전경에서 실행시키거나 bg지령으로 배경에서 실행시킬수 있다. bg지령은 지령의 뒤에 "&"를 붙여 준것과 같다. 그 지령은 중단된 위치에서 시작된다. fg지령 또는 bg지령을 줄 때 일감번호를 주지 말아야 한다. 왜냐하면 표준적으로 마지막일감에서 규정된 조작

을 수행하는것이기때문이다. 이 실례에서 마지막일감번호는 4이다. 이것은 다른 일감들은 보다 낮은 번호를 가지고 실행되고 있다는것을 의미한다. 모든 일감들의 목록과 그것들의 상태를 얻기 위하여 jobs지령을 리용한다.

fg지령뒤에 "%"와 일감번호를 붙여 주어 그 지령이 전경에서 실행되게 하거나 bg지령뒤에 "%"와 일감번호를 붙여 주어 그 지령이 배경에서 실행되게 할수 있다. 일감을 종결시키려면 kill지령에 "%"와 일감번호를 붙여 주면 된다. 다음의 실례는 jobs지령으로 모든 일감들을 렬거하고 kill지령으로 일감1과 일감 2를 제거하며 일감 3을 배경에서 실행시키는것을 보여 준다.

```
# jobs
[1]         Stopped ls  -alR
[2]  -      Stopped run_audit
[3]  +      Stopped run_check
# kill  %1
[1]         Stopped
# kill  %2
[2]         Stopped
# bg  %3
#
```

일감 1과 일감 2를 제거하고 일감 3을 배경에서 실행시키면 다른 작업을 수행할수 있도록 입력재촉문이 즉시 표시된다.

umask와 허락

쉘프로그람작성기술에 앞서 파일들에 대한 허락(Permission)들과 그것들을 umask로 설정하는 방법을 취급한다. 이것은 모든 사람들이 리용할수 있는 쉘프로그람들과 제한된 수의 사용자들(가능하게는 체계관리자)만이 리용할수 있는 쉘프로그람들을 작성하려고 하는 경우에 중요한 문제로 제기된다. umask는 새 파일들과 등록부들에 대한 허락을 규정하는데 리용된다.

다음의 실례를 고찰하자. 이 실례에서는 ls -l의 별명 ll을 리용한다.

```
sys1 1: ll script1
-rwxr-xr-x  1  marty  users  120  Jul  26  10:20  script1
```

이 파일에 대한 **접근권한**(access right)은 지령 ll을 줄 때 읽기(r), 쓰기(w), 실행(x)의 위치에 의해 정의된다. 그림 9-2는 이 파일에 대한 3가지 접근권한들의 그룹들을 보여 준다.

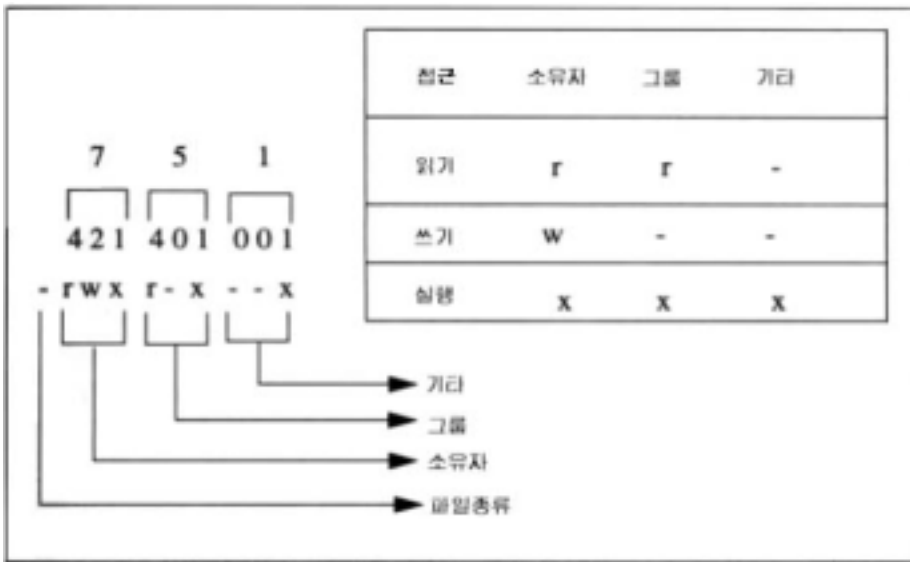



그림 9-2. 파일허락의 실례

파일의 소유자는 자기의 파일에 대한 **읽기허락**(Read Permission), **쓰기허락**(Write Permission), **실행허락**(Execute Permission)을 가진다. 사용자가 속하는 그룹과 기타 다른 사람들은 읽기허락과 실행허락을 가진다. 파일에 대한 허락들은 매 마당의 8진합에 의해 규정된다. 이 경우에는 755이다.

새로운 쉘스크립트 또는 임의의 새로운 파일을 만들면 어떤 문제가 생기며 어떤 허락설정들이 존재하는가 하는 문제가 제기된다. 쉘스크립트를 실행시키려면 이 파일에 대한 실행허락이 필요된다. umask를 리용하여 모든 새로운 파일들과 등록부들에 대한 표준값을 정의할수 있다.

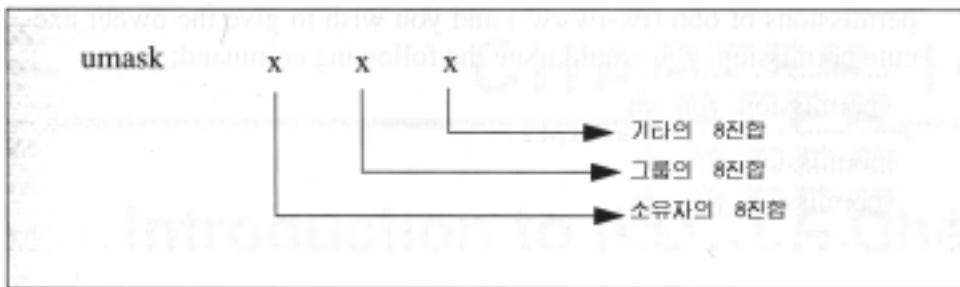



그림 9-3. umask마당들

다음의 지령으로 umask를 볼수 있다.

```
sys1 2: umask
```

.cshrc에서 umask를 설정하여 새로운 파일들과 등록부들에 대한 허락을 정의할수 있다. umask로 접근을 불가능하게 하려면 3개의 8진마당들을 리용한다. 그 마당들은

그림 9-3에서 보여 주는바와 같이 사용자와 그룹들 그리고 기타 다른 사람들에 대한 접근코드들의 합이다.

umask마당의 부정은 기정설정과 론리적되여 umask를 변경시킨다. 그림9-4에서 "group"과 "other"에 대한 쓰기허락을 제거하려면 아래에서 보여 주는것처럼 022의 umask를 할당하여야 한다.

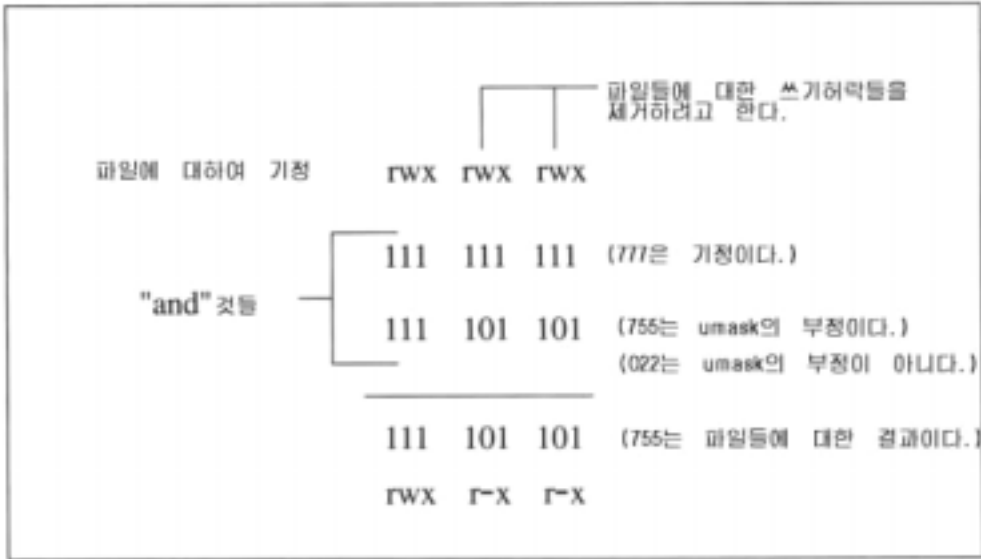



그림 9-4. umask실례

이 실례에서 umask 022는 파일허락을 755로 변경시킨다. 쉘스크립트를 위한 하나의 새로운 파일(script2)을 창조하면 그것을 지령 chmod로 실행가능하게 하여야 한다. 666(rw-rw-rw-)의 허락을 가지는 파일에 소유자실행허락을 주려면 다음의 지령을 주어야 한다.

Sys1 3:**chmod 766 script2**

제１０장. Korn쉘에 대한 개괄

각이한 쉘

대부분의 UNIX변종들에는 여러개의 쉘들이 있다. 쉘을 통하여 체계를 쉽게 리해할수 있기때문에 사용자들에게 있어서 쉘은 매우 중요하다. 쉘들에서 지령들을 주고 사용자환경들을 조종하며 지령파일들과 쉘프로그람들을 기입하는 등 UNIX변종들에서 쉘들을 리용하는 방법은 류사하다. 쉘에서는 대체로 많은 시간이 소비되기때문에 여러개의 각이한 쉘들을 리해하고 그것들중의 하나를 선택해야 한다. Bash쉘과 같은 특수한 쉘들은 UNIX변종들에서 그리 차이나지 않는다. 쉘들은 자체의 고유한 특징들을 가진다. 일반적으로 사용자들은 어느 한개 쉘을 특별히 좋아 하게 된다. 모든 쉘들은 기능적으로 류사하며 그것을 안 다음에는 리용하기가 재미 있다. 대부분의 UNIX변종들에서 체계관리자는 사용자들을 등록할 때 여러개의 각이한 쉘들중에서 하나를 선택할수 있으므로 사용자들이 실행시키는 쉘들을 유연하게 처리할수 있다. 일반적으로 체계관리자들은 체계관리를 쉽게 하기 위하여 사용자들이 동일한 쉘을 리용할것을 요구한다. 그러나 체계관리자들은 사용자가 체계에 있는 특정한 쉘을 리용해야 할 리유가 있으면 그에 대한 사용자의 요구를 들어 주기도 한다. 이 장에서는 Korn쉘을 취급한다. Bash쉘은 앞 장에서 취급되였다. 다음 장에서는 C쉘을 취급한다.

Korn쉘에 대한 개괄

대부분의 UNIX변종들에는 여러개의 쉘들이 있다. 리용하기 쉬운것으로 하여 많은 체계관리자들은 새로운 사용자들을 위하여 Korn쉘을 설치해 준다. Korn쉘은 Bourne 쉘로부터 유도되였으며 그것과 동일한 기능들을 많이 가지고 있다. ksh는 Korn쉘기능을 보장하는 UNIX체계상에서 실행되는 프로그람이다. 그것을 흔히 K쉘이라고 부른다. 이 장에서는 ksh가 리용된다. ksh는 다음과 같은 3가지 방식으로 리용할수 있다.

- 대화적으로 지령행에 지령들을 입력한다.
- 흔히 리용되는 지령모임들을 지령파일에 그룹화하고 그 파일의 이름을 입력하여 실행시킨다.
- 쉘의 구조화된 프로그람작성기술을 리용하여 Korn쉘프로그람들을 작성한다.

이 3가지 기술들은 자주 리용되는 순서로 렬거되였다. 우선 체계에 가입하고 대화적 지령들을 리용한다. 그 다음 흔히 리용되는 지령들을 그룹화하고 그것을 하나의 지령으로 실행시킨다. 끝으로 쉘스크립트들을 창조한다.

그러므로 여기서는 Korn쉘의 개념들을 우에서 렬거된 순서로 서술한다. Korn쉘의

지령파일들과 프로그람작성개념들은 "쉘프로그람작성"장의 일부분으로 취급된다.

이 장에 있는 대부분의 실례들은 Solaris체계의것이다. 이 장에서 취급되는 체계와 류사한 다른 체계들에서도 Korn쉘을 설치하고 리용할수 있다. 쉘의 설치가 체계관리자에 의하여 수행되므로 이 장에서 취급되지 않는 다른 Korn쉘에서는 차이가 있을수 있다. 그러나 일반적으로 Korn쉘의 조작은 모든 체계들에서 류사하다.

시동파일

체계에 가입한 다음 사용자가 수행하는 첫 작업은 **입력재촉문**(Prompt)에 따라 지령을 주는것이다. 처음으로 주어야 할 지령은 ls -al이다. 이 지령이 실행되면 다음과 같은 파일목록이 생겨 난다.

```
martyp $ ls -al
total 22
drwxr-xr-x   2  martyp  staff   512  Mar  15  11:37  .
drwxrwxr-x   4    root    sys   512  Mar   4  09:24  ..
-rw-r--r--   1  martyp  staff   124  Mar  15  11:36  .cshrc
-rw-r--r--   1  martyp  staff   562  Mar   4  09:24  .profile
-rw-------   1  martyp  staff  7056  Apr   6  07:24  .sh_history
martyp $
```

이 파일목록은 짧다. .profile과 .cshrc와 같은 일부 파일들은 각각 ksh와 C쉘을 위한 시동파일이다. 체계에 가입하면 쉘을 실행시킨 다음 홈등록부에 있는 국부적인 시동파일을 실행시킨다. 이 경우에 홈등록부에는 매우 적은 내용을 담고 있는 최소 ksh시동파일이 있다. 사용자를 위한 모든 시동절차들은 체계륜곽파일 /etc/profile에 들어 있다. /etc/profile을 읽어 보면 체계관리자가 사용자들을 위해 무엇을 설치하였는가를 알수 있다. 사용자는 다양한 기능을 포함하도록 국부적인 .profile을 변경시킬수 있다. 국부적인 .profile은 체계에 가입할 때 /etc/profile 다음에 곧 실행된다.

ksh가 가입에 필요되는 모든 실행스크립트들을 실행한후에 환경은 사용자를 위하여 설치된다. 국부적인 .profile은 재촉통보문을 설치하는것외에 다른것을 수행하지 않는다.

환경파일(Environment File)이라고 부르는 파일이 있는데 이 파일의 이름은 환경변수 **ENV**에 의하여 정의된다. 이 파일은 보통 홈등록부에 있는 .kshrc이다. 환경파일을 리용하면 부분프로세스에 전달될 선택항목, 별명 그리고 기타 정보들을 정의할수 있다. 표준적으로는 이미 존재하는 환경변수들이 부분프로세스에 전달된다. 환경파일은 기타 정보를 전달하는데도 리용된다.

/etc/profile, .profile, .kshrc를 포함하는 ksh의 시동파일들을 살펴 보아야 한다.

아래에서는 시동프로그람의 부분으로 이루어 진 특정한 ksh기능을 보여 준다.

리력파일

ksh는 실행된 지령들의 **리력목록**(History List)을 가지고 있다. 지령을 다시 주거나 앞서 실행된 지령을 보려고 하는 경우에는 리력목록을 리용할수 있다. 표준적으로는 홈등록부에 있는 .sh_history가 리력파일로 리용된다. 체계관리자가 ksh홈등록부를 창조할때 대체로 이 파일은 만들어 지며 기정리력파일로 리용된다.

대부분의 체계들은 가장 최근에 실행된 128개의 지령들을 보관한다. 리력목록에 포함될 지령들의 개수를 규정해 줄수 있다. 아래의 행에서는 리력목록을 20으로 설정해 준다.

HISTSIZE=200

사용자들은 이것을 자기의 홈파일 .profile에 기입한다. .profile에 이 행을 첨가하면 리력목록은 200으로 설정된다.

리력목록으로부터의 재호출

아래의 실례에서 보여 주는바와 같이 가장 최근에 실행된 지령들과 그 지령들에 대응하는 행번호를 history지령으로 볼수 있다.

```
martyp $ history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
martyp $
```

이 실례에서 지령번호 127 은 exit지령 다시 말하여 작업을 끝내는 지령이다. 지령번

호 128 은 다음 작업을 시작할 때 주는 env지령이다.

선택항목 -n을 주면 아래의 실례에서 보여 주는바와 같이 행번호가 없이 리력목록을 인쇄할수 있다.

```
martyp $ history -n
          history 128 | more
          history 200
          alias
          history 1
          inv
          env
          env | more
          history 1
          more .profile
          env | grep HIS
          exit
          env
          env | grep HIS
          more .profile
          history
          history -n
martyp $
```

리력목록에 있는 지령목록을 전부 보려면 다음의 지령을 주어야 한다.

```
martyp $ history 0
5         env
6         more .profile
7         set
8         env | grep CDP
9         env | grep cdp
10        echo $SHELLL
11        echo $SHELL
12        more .profile
13        ll
14        ls -al /etc/profile
15        more /etc/profile
16        more /etc/profile | grep PS
17        exhoecE
```

```
18        ECHO
19        echo  $ENV
20        env  |  more
21        echo  $ENV
22        env  |  more
23        env  |  grep  ENV
24        env  |  grep  env
25        more  /usr/bin/env
26        ls  -al
27        more  /etc/.kshrc
28        find  /  -name  .kshrc
29        more  /etc/passwd
30        more  /etc/passwd  |  grep  /home
31        ll  /home/oracle
32        ls  -al  /home/oracle
33        ll  /home
34        ls  -al  /home
35        ls  -al  /home/ptc-nfs
36        ls  -al  /home/ptc-nfs  |  more
37        ls  -al  /root
38        cd  /root/users
39        ls  -al
40        ls  -al    verasu
41        is  -al    rodtsu
42        cd         rodtsu
43        more  .kshrc
44        more  .env
45        ls
46        ls  -1
47        ls  -a
48        more     .kshrc
49        cd  ..
50        ls  -al
51        ls  -al  |  more
52        exit
53        ls  -al
54        more  .sh*
55        history
56        env  |  grep  his
```

```
57    env | grep HIS
58    env | grep IS
59    env | more
60    env | grep FC
61    env | grep EDI
62    history
63    echo $HISTIZE
64    echo $HISTSIZE
65    env | more
66    ls -al
67    history 128
68    history
69    history 1 104
70    exit
71    alias
72    more /etc/profile | grep alias
73    ll
74    la -al
75    ls -al
76    more .profile
77    ls -al
78    more .cshrc
79    alias ll="ls -al"
80    aliase
81    alias
82    ll
83    exit
84    alias
85    alias
86    history
87    fc -1
88    man fc
89    man fc
90    man fc
91    alias
92    ps
93    ps -ef
94    ps -ef 1
95    ps
```

```
96      ls -alF
97      ls -al
98      alias ls="ls -al'
99      alias
100     ls
101     alias
102     unalias ls
103     ls
104     ls -al
105     alias
106     history
107     alias
108     unalias history
109     history
110     fc -1
111     history
112     alias hisotry="fc -l"
113     history
114     alias history="fc -l"
115     history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
132     history 0
martyp $
```

history 0은 제일 첫 지령인 0번 지령부터 현재 지령까지 인쇄한다. 128개의 지령들이 기정으로 보관되여 있기때문에 가장 오래된 지령들 0−4는 리력목록에서 제거된다. 이 사실은 우리가 지령번호 0으로부터 목록이 현시될것을 요구하였지만 지령번호 5에서부터 현시되였다는것을 의미한다.

지령번호100부터 현재까지의 목록을 현시하려면 다음의 지령을 주어야 한다.

```
martyp $ history 100
100     ls
101     alias
102     unalias ls
103     ls
104     ls -al
105     alias
106     history
107     alias
108     unalias history
109     history
110     fc -l
111     history
112     alias hisotry="fc -l"
113     history
114     alias history="fc -l"
115     history
116     history 128
117     history 128 | more
118     history 200
119     alias
120     history 1
121     inv
122     env
123     env | more
124     history 1
125     more .profile
126     env | grep HIS
127     exit
128     env
129     env | grep HIS
130     more .profile
131     history
```

```
132        history 0
133        history
134        history -n
135        history 100
martyp $
```

현재지령과 그앞의 20개 지령들을 렬거하려면 다음의 지령을 주어야 한다.

```
martyp $ history -20
116        history 128
117        history 128 | more
118        history 200
119        alias
120        history 1
121        inv
122        env
123        env | more
124        history 1
125        more .profile
126        env | grep HIS
127        exit
128        env
129        env | grep HIS
130        more .profile
131        history
132        history 0
133        history
134        history -n
135        history 100
136        history -20
martyp $
```

-20은 현재 지령으로부터 뒤로 20개 지령만큼 이동하게 한다. 이 경우에 history는 현 지령을 목록현시를 시작할 위치로 리용하고 있다. 아래의 실례에서 보여 주는바와 같이 범위 -1 -20을 규정해 주면 현 지령의 앞에 있는 마지막20개 지령들을 렬거한다.

```
martyp $ history -1 -20
136        history -20
135        history 100
```

```
134         history -n
133         history
132         history 0
131         history
130         more .profile
129         env | grep HIS
128         env
127         exit
126         env |grep HIS
125         more .profile
124         history 1
123         env | more
122         env
121         inv
120         history 1
119         alias
118         history 200
117         history 128 | more
martyp $
```

이 실례에서 현재 지령 history -1 -20은 목록에 없다. 왜냐하면 -1이 앞의 지령에서부터 시작하여 렬거되기때문이다. 이 지령은 렬거해야 할 지령들의 범위를 지적하는 효과적인 방법이다.

아래의 실례에서 보여 주는바와 같이 -20을 앞에, -1을 뒤에 규정하면 마지막 20개 지령들의 순서를 바꿀수 있다.

```
martyp $ history -20 -1
118         history 200
119         alias
120         history 1
121         inv
122         env
123         env | more
124         history 1
125         more .profile
126         env | grep HIS
127         exit
128         env
129         env |grep HIS
```

```
130         more  .profile
131         history
132         history  0
133         history
134         history  -n
135         history  100
136         history  -20
137         history  -1  -20
martyp  $
```

어느 한 지령을 주었을 때부터 현 지령까지의 목록도 현시할수 있다. 아래의 실례는 마지막지령 fc로부터 현 지령까지의 목록을 현시한다.

```
martyp  $ history  fc
110         fc  -1
111         history
112         alias  hisotry="fc  -l"
113         history
114         alias  history="fc  -l"
115         history
116         history  128
117         history  128  |  more
118         history  200
119         alias
120         history  1
121         inv
122         env
123         env  |  more
124         history  1
125         more  .profile
126         env  |  grep  HIS
127         exit
128         env
129         env  |  grep  HIS
130         more  .profile
131         history
132         history  0
133         history
134         history  -n
```

```
135        history  100
136        history  -20
137        history  -1  -20
138        history  -20  -1
139        history
140        history  grep
141        history  env
142        history
143        history  fc
martyp  $
```

아래의 실례에서 보여 주는바와 같이 선택항목 -r를 리용하여 이 목록을 뒤집을수 있다.

```
martyp  $  history  -r  fc
144        history  -r  fc
143        history  fc
142        history
141        history  env
140        history  grep
139        history
138        history  -20  -1
137        history  -1  -20
136        history  -20
135        history  100
134        history  -n
133        history
132        history  0
131        history
130        more  .profile
129        env  |  grep  HIS
128        env
127        exit
126        env  |  grep  HIS
125        more  .profile
124        history  1
123        env  |  more
122        env
121        inv
```

```
120         history  1
119         alias
118         history  200
117         history  128  |  more
116         history  128
115         history
114         alias  history="fc  -l"
113         history
112         alias  hisotry="fc  -l"
111         history
110         fc  -l
martyp  $
```

이와 같이 리력목록을 리용하는것은 Korn쉘의 중요한 특성이다. 입력오유를 포함하여 실행된 모든 지령들을 임의의 형식으로 볼수 있다. 다음항목에서는 리력목록에 있는 지령들을 재호출하여 리용하는 방법에 대하여 취급한다.

r로 지령들을 재실행

지령 r로 어느 한 지령을 다시 줄수 있다. 마지막지령을 다시 주려면 단순히 r를 입력하면 된다. 아래의 실례에서는 리력목록에 있는 마지막5개 지령을 보기 위하여 지령 history 5 를 주었다. 그다음 마지막지령을 다시 주기 위하여 r를 입력하였으며 다시 마지막 5개 지령들을 렬거하였다.

```
martyp  $  history  -5
301         whoami
302         pwd
303         more  /etc/passwd
304           ps  -efl
305         cat  . profile
306         history  -5
martyp  $  r
history  -5
302         pwd
303         more  /etc/passwd
304           ps  -efl
305         cat  . profile
306         history  -S
307         history  -5
martyp  $
```

r와 공백, 실행시키려는 지령번호를 입력하여 특정한 지령번호를 다시 줄수 있다. 아래의 실례에서 리력지령번호 302를 주었다.

martyp $ **history**
294 history -5
295 whoami
296 pwd
297 more /etc/passwd
298 ps -efl
299 cat /etc/profile
300 history 5
301 whoami
302 pwd
303 more /etc/passwd
304 ps -efl
305 cat .profile
306 history -5
307 history -5
308 history -5
309 history
martyp $ **r 302**
pwd
/home/martyp
martyp $

실행시키려는 지령의 이름도 줄수 있다. 마지막지령 fc를 다시 주려고 한다고 하자. 아래의 실례에서 보여 주는바와 같이 지령 r와 함께 문자 "f"를 주면 f로 시작되는 마지막지령이 다시 실행된다.

martyp $ **history**
308 history -5
309 history
310 pwd
311 whoami
312 pwd
313 cat /etc/profile
314 ls
315 fc -1
316 env

317　　　who
318　　　cat /etc/passwd
319　　　ls -al
320　　　history
321　　　more profile
322　　　set
323　　　history
martyp $ **r f**
fc -1
309　　　history
310　　　pwd
311　　　whoami
312　　　pwd
313　　　cat /etc/profile
314　　　ls
315　　　fc -1
316　　　env
317　　　who
318　　　cat /etc/passwd
319　　　ls -al
320　　　history
321　　　more .profile
322　　　set
323　　　history
324　　　fc -1
martyp $

지령 r로는 일정한 치환도 할수 있다. 먼저 지령 vi를 주고 다음에 편집하려는 다른 파일이름을 규정하며 vi를 다시 실행시키려는 경우에 새 파일이름으로 치환할수 있다.

아래의 실례는 이미 편집되여 있는 파일 .profile을 새로 편집하려고 하는 파일이름 .kshrc로 교체하기 위하여 앞에서 준 지령 vi를 다시 실행시키는것을 보여 주고 있다.

martyp $ **history**
316　　　env
317　　　who
318　　　cat /etc/passwd
319　　　ls -al
320　　　history
321　　　more .profile
322　　　set

```
323          history
324          fc -1
325          vi .profile
326          set
327          ls -al
328          env
329          who
330          more /etc/passwd | grep marty
331          history
martyp $ r vi .profi1e=.kshrc
martyp $ history
318          cat /etc/passwd
319          ls -al
320          history
321          more .profile
322          set
323          history
324          fc -1
325          vi .profile
326          set
327          ls -al
328          env
329          who
330          more /etc/passwd | grep marty
331          history
332          vi .kshrc
333          history
martyp $
```

이 지령은 지령행에서 .profile을 .kshrc로 치환한다.

앞에서 실행된 한 지령을 다시 호출하고 vi의 편집지령들을 리용하여 지령행을 편집할수 있다.

다음항목에서는 지령행편집을 취급한다.

vi의 지령문을 리용한 지령들의 꺼내기

편집기 vi 또는 emacs로 리력목록에 있는 항목들을 편집할수 있다. 이 장에서는 UNIX에서 가장 널리 리용되는 편집기 vi를 취급한다. 그러나 이 장에서 보여 주는 기능들은 모두 편집기 emacs에서도 수행될수 있다.

vi를 리용하여 리력목록을 편집하는 방법과 파일을 편집하는 방법은 같다. 여기서는 vi의 지령들을 리용하여 리력목록으로부터 지령을 꺼내는 방법과 그것을 지령행에 가져온후 그 지령을 편집하는 방법을 취급한다. 지령이 편집된후에는 <enter>건을 눌러 그 지령을 수행하고 그것을 지령목록의 맨 아래에 넣는다.

편집기는 보통 국부적인 .profile 또는 /etc/profile에 설정되여 있다. 이 파일을 보고 체계관리자가 어떤 편집기를 설정하였는가 하는것을 알수 있다. 체계상에서 지령 env와 set를 주면 둘 다 EDITOR=vi를 제시한다.

우선 이미 실행된 몇개의 지령들을 vi를 리용하여 꺼내는 방법을 보자. 여기에 있는 실례들은 "vi"장과 "vi고속참조카드"부분에 서술된 기능들을 리용한다. 지령 k, j, G와 탐색지령들은 파일을 편집하기 위하여 vi를 리용할 때와 동일한 방식으로 지령행에서 동작한다.

<ESC>건과 어느 한 지령을 입력하여 ksh의 다양한 기능들을 수행시킬수 있다.

아래의 실례는 리력목록을 생성하고 선행한 지령을 재호출하기 위하여 escape k를 주는것을 보여 준다.

```
martyp $ history
125        man fc
126        set | more
127         exit
128        history
129        k
130        env | more
131        set | more
132        more .profile
133        more .profile | grep rof
134        more .profile | grep vi
135        more /etc/profile | grep vi
136        set | grep edit
137        set | grep vi
138        env | grep vi
139        history
140        who
martyp $ history
```

escape k를 주면 마지막에 실행된 지령(이 경우에는 140행에 있는 지령) who가 재호출된다. k를 누를 때마다 바로 앞의 ksh지령이 꺼내진다. 이것은 escape와 -건(미누스건)을 누르는것과 동일하다. 리력목록에서 되돌아 가려고 하는 지령들의 개수를 escape nk로 규정해 줄수 있다. 여기서 n은 리력목록에서 되돌아 가려고 하는 지령들의 개수이다.

아래의 실례에서는 리력목록에서 다섯번째 지령을 꺼내기 위하여 escape 5k를 준다.

```
martyp $ history
125	man fc
126	set | more
127	exit
128	history
129	k
130	env | more
131	set | more
132	more .profile
133	more .profile | grep rof
134	more .profile | grep vi
135	more /etc/profile | grep vi
136	set | grep edit
137	set | grep vi
138	env | grep vi
139	history
140	who
martyp $ set | grep edit
```

escape 5k를 주면 리력목록에서 거꾸로 다섯번째 지령인 136행의 지령이 꺼내진다. 이것은 escape 5-를 준것과 동일하다.

k를 리용하여 리력목록에서 뒤로 이동한것과 마찬가지로 j를 리용하여 리력목록에서 앞으로 이동할수 있다. 아래의 실례에서 보여 주는바와 같이 리력목록에서 뒤로 다섯개 지령을 이동하여 136행으로 가기 위하여 escape 5k를 주고 앞으로 세개 지령을 이동하여 139행으로 가기 위하여 escape 3j를 줄수 있다.

```
martyp $ history
125	man fc
126	set | more
127	exit
128	history
129	k
130	env | more
131	set | more
132	more .profile
133	more .profile | grep rof
134	more .profile | grep vi
135	more /etc/profile | grep vi
136	set | grep edit
```

```
137         set | grep vi
138         env | grep vi
139         history
140         history
martyp $ who
```

escape 3j를 주면 앞으로 이동하여 139행에로 간다. 이것은 escape 3-를 준것과 같다.

escape G를 주면 리력목록에서 가장 오랜 지령에로 되돌아 가며escape nG를 주면 리력목록에 있는 특정한 지령번호에로 되돌아 간다. 여기서 n은 리력목록안의 지령번호이다. 아래의 실례는 리력목록안의 136행으로 가기 위하여 escape 136G를 준것을 보여 준다.

```
martyp $ history
125         man fc
126         set | more
127            exit
128         history
129         k
130         env | more
131         set | more
132         more .profile
133         more .profile | grep rof
134         more .profile | grep vi
135         more /etc/profile | grep vi
136         set | grep edit
137         set | grep vi
138         env | grep vi
139         history
140         who
martyp $ set | more
```

리력목록에 있는 행들을 탐색에 기초하여 재호출할수도 있다. 실례로 profile이 들어있는 리력목록으로부터 가장 최근에 리용된 지령을 재호출하려는 경우에 아래의 실례에서 보여 주는 결과를 얻으려면 /profile을 주어야 한다.

```
martyp $ history
125         man fc
126         set | more
127            exit
128         history
```

```
129        k
130        env | more
131        set | more
132        more .profile
133        more .profile | grep rof
134        more .profile | grep vi
135        more /etc/profile | grep vi
136        set | grep edit
137        set | grep vi
138        env | grep vi
139        history
140        who
martyp $ more /etc/profile | grep vi
```

profile이 제일 마지막으로 출현한 행은 135행이다. vi에서의 많은 탐색들은 지령행에서 작업한다. "vi"장과 "vi고속참조카드"에서는 지령행에서 리용할수 있는 많은 탐색정보들을 담고 있다.

vi의 지령문을 리용한 지령행의 편집

편집하려는 지령을 꺼냈거나 입력하였다면 vi의 지령들을 리용하여 그것을 지령행에서 편집할수 있다.

ksh는 리력목록으로부터 한 지령을 꺼내면 곧 입력방식에로 넘어 간다. 반대로 vi는 조종방식에로 넘어 가 파일을 편집한다. 리력목록으로부터 한 지령을 꺼낸후에 ksh에서 vi를 리용할 때 조종방식으로 넘어 가기 위하여서는 escape를 눌러야 한다.

이제 리력목록으로부터 꺼낸 지령들을 간단히 변경해 보자. escape 286G로 리력목록으로부터 행번호 286을 재호출한다. 이 지령을 재호출한 다음 i cat로 그 지령의 앞에 cat를 삽입한다. i는 현 유표위치의 왼쪽에 본문을 삽입하기 위한것이다.

아래의 실례는 286행을 꺼내고 cat를 삽입한 결과를 보여 준다.

```
martyp $ history
285        history
286        more /etc/profile | grep vi
287        who
288        whoami
289        who
290        ls -al
291        alias
292        pwd
```

```
293      ls -al /home
294      cat . profile
295      history
296      vi . profile
297      cat /etc/passwd
298      cat /etc/passwd | grep donna
299      pwd
300      history
martyp $ cat more /etc/profile | grep vi
```

i cat는 행의 맨 앞에 cat를 삽입하였다.

아래의 실례에서 보여 주는바와 같이 286행을 다시 호출하고 그 행의 맨뒤에 본문을 추가하기 위하여 A를 리용할수 있다.

```
martyp $ history
285      history
286      more /etc/profile | grep vi
287      who
288      whoami
289      who
290      ls -al
291      alias
292      pwd
293      ls -al /home
294      cat profile
295      history
296      vi . profile
297      cat /etc/passwd
298      cat /etc/passwd | grep donna
299      pwd
300      history
martyp $ more /etc/profile | grep vieditor
```

이 실례에서는 재호출한 지령행의 맨끝에 단어 editor를 삽입하기 위하여 A editor를 리용하였다.

이 행을 재호출하고 아래의 실례에서 보여 주는바와 같이 령 cw cat를 주어 그 행의 앞에 있는 more를 cat로 교체한다.

```
martyp $ history
285         history
286         more /etc/profile | grep vi
287         who
288         whoami
289         who
290         ls -al
291         alias
292         pwd
293         ls -al /home
294         cat .profile
295         history
296         vi .profile
297         cat /etc/passwd
298         cat /etc/passwd | grep donna
299         pwd
300         history
martyp $ cat /etc/profile | grep vi
```

교체하지 않고 단어 more를 삭제하려면 아래의 실레에서 보여 주는바와 같이 286행을 꺼내고 dw를 준다.

```
martyp $ history
285         history
286         more /etc/profile | grep vi
287         who
288         whoami
289         who
290         ls -al
291         alias
292         pwd
293         ls -al /home
294         cat profile
295         history
296         vi profile
297         cat /etc/passwd
298         cat /etc/passwd | grep donna
299         pwd
300         history
martyp $ /etc/profile | grep vi
```

가장 최근에 실행된 지령을 무효로 하려면 u를 준다. 아래의 실례는 지령행에서 more를 제거하기 위하여 리용된 dw를 무효로 하는것을 보여 준다.

```
martyp $ history
285        history
286        more /etc/profile | grep vi
287        who
288        whoami
289        who
290        ls -al
291        alias
292        pwd
293        ls -al /home
294        cat .profile
295        history
296        vi .profile
297        cat /etc/passwd
298        cat /etc/passwd | grep donna
299        pwd
300        history
martyp $ more /etc/profile | grep vi
```

u를 주면 dw로 제거하였던 more가 다시 생겨 난다.

여기서 취급된 vi의 지령들뿐아니라 지령행에서의 첨가, 삭제, 변경, 탐색, 교체, 복사, 무효를 위한 지령들도 리용할수 있다.

Korn쉘에서의 별명들

별명(alias)은 자주 리용되는 지령 또는 지령렬에 대하여 선택하는 이름이다. 많은 별명들이 이미 정의되여 있다.

아무런 인수도 없이 지령 alias를 주면 모든 별명들이 렬거된다. 이 목록에는 이미 설정된 별명들뿐아니라 사용자가 설정한 별명들도 포함된다.

아래의 지령은 현재 작업하고 있는 체계상에 미리 설정되여 있는 별명들을 보여 준다.

```
martyp $ alias
autoload= ' typeset -fu '
command= ' command '
functions= ' typeset -f '
```

```
history='fc -l'
hyperl=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptcl;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i'
local=typeset
nohup='nohup'
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
martyp $
```

이 별명들중의 많은것들은 Korn쉘의 어느 정도 개선된 내용인 일감조종에서 리용될수 있다.

다른것들은 즉시 리용할수 있다. 실례로 별명 history는 .sh_history에 있는 지령들을 렬거하면서 매 항목들앞에 행번호를 붙인다. 별명 r는 리력목록에 있는 마지막지령이 재실행되도록 한다.

미리 설정된 별명들만을 리용하는데는 한계가 있다. 사용자들은 자기자신의 별명들을 설정할수 있다.

아래의 실례는 새 별명을 설정하고 그 별명이 실지로 설정되였는가를 보기 위하여 별명들의 목록을 생성하며 별명이 실행되는것을 보여 준다.

martyp $ **alias ls="ls -al"**
martyp $ **alias**

```
autoload='typeset -fu'
command='command '
functions='typeset -f '
history='fc -l'
hyperl=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptcl;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i'
local=typeset
ls='ls -al'
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
```

```
suspend='kill -STOP $$'
martyp $ ls
total 26
drwxr-xr-x    2  martyp  staff  512   Mar 15  11:37  .
drwxrwxr-x    4  root    sys    512   Mar 4   09:24  ..
-rw-r--r--    1  martyp  staff  124   Mar 15  11:36  .cshrc
-rw-r--r--    1  martyp  staff  562   Mar 4   09:24  .profile
-rw-------    1  martyp  staff  9058  Apr 10  06:58  .sh_history
martyp $
```

첫 지령은 ls를 입력하자마자 ls -al을 실행하는 별명을 설정한다. 새 별명이 실지로 별명들의 목록에 나타나는가를 보기 위해 지령 alias를 주었다. 별명들을 자모순서로 볼때 실지로 local뒤에 그리고 nohup앞에 새 별명이 놓여 있는것을 볼수 있다. 마지막지령은 ls의 실행을 보여 준다. 이것은 지령 ls -al을 실행시켰을 때와 같은 결과를 준다.

별명을 설정한 후 그것을 작업전기간 유지하지 않아도 된다. 어느 한 별명이 좋지 않으면 지령 unalias를 리용하여 그것을 제거할수 있다.

unalias의 동작을 보기 위하여 별명들의 목록을 다시 생성하고 별명 history를 없애기 위하여 unalias을 리용하며 그것이 실지로 제거되였는가를 보기 위하여 지령 history를 실행시키며 별명 history가 대응된 지령 fc -l을 실행시키자.

```
martyp $ alias
autoload='typeset -fu'
command='command '
functions='typeset -f '
history='fc -l'
hyper1=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc1;export HHLIC'
hyper36=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc36;export HHLIC'
hyper83=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc83;export HHLIC'
hyper95=HHLIC='/opt/local/bristol/hyperhelp/licenses/hpptc95;export HHLIC'
integer='typeset -i '
local=typeset
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
martyp $ unalias history
martyp $ history
ksh: history: not found
martyp $ fc -l
```

```
131       ps
132       ls -alF
133       ls -al
134       alias ls="ls -al"
135       alias
136       ls
137       alias
138       unalias ls
139       ls
140       ls -al
141       alias
142       history
143       alias
144       unalias history
145       history
146       fc -l
martyp $
```

지령 unalias history을 실행시킨 다음 history지령을 주면 ksh는 그것을 찾지 못하며 통보문 history: not found를 출력한다. fc -l를 실행시키면 가장 최근에 실행된 지령들의 목록이 대응하는 행번호와 함께 출력된다. 이것은 별명 history를 실행시키는것이 실지로는 지령 fc -l을 실행시키는것이라는것을 보여 준다.

지령과 경로의 완성

ksh는 때때로 사람이 무엇을 생각하고 있는가를 알아 차린다. 지령 또는 경로이름의 일부분만을 입력하고 escape건을 누르면 그 지령이 완성된다. 실례로 현 체계의 실행준위를 보기 위하여 지령 uname을 주려고 할 때 그 지령이름을 잘 모른다고 하면 "un"과 escape건 그리고 "="건을 누르면 그 지령이 완성된다.

아래의 실례에서는 /sbin등록부에로 변경하고 그 등록부안의 내용들을 렬거한 다음 **un**escape=를 입력한다.

```
martyp $ cd /sbin
martyp $ ls -al
total 11704
drwxrwxr-x    2  root   sys       512  Nov  27  14:55   .
drwxr-xr-x   31  root   root     1024  Apr  19  03:24   ..
-r-xr-xr-x    1   bin   bin    200356  Sep   1   1998   .autopush
lrwxrwxrwx    1  root   root       21  Nov  27  14:55   .bpgetfile-> ../usr/sbin/e
```

```
-r-xr-xr-x   1  bin   bin   470436  Sep   1  1998  dhcpagent
-r-xr-xr-x   1  bin   bin   433064  Sep   1  1998  dhcpinfo
-r-xr-xr-x   1  bin   bin   253664  Sep   1  1998  fdisk
-r-xr-xr-x   1  bin   bin   762816  Sep   1  1998  hostconfig
-r-xr-xr-x   1  bin   bin   535900  Sep   1  1998  ifconfig
-r-xr-xr-x   1  root  sys   516484  Sep   1  1998  init
-r-xr-xr-x   2  bin   root  257444  Sep   1  1998  jsh
-r-xr-xr-x   1  bin   bin   224596  Sep   1  1998  mount
-r-xr-xr-x   1  root  sys     6935  Jan   1  1970  mountall
-rwxr--r--   3  root  sys     2689  Jan   1  1970  rc0
-rwxr--r--   1  root  sys     2905  Jan   1  1970  rc1
-rwxr--r--   1  root  sys     2491  Jan   1  1970  rc2
-rwxr--r--   1  root  sys     1948  Jan   1  1970  rc3
-rwxr--r--   3  root  sys     2689  Jan   1  1970  rc5
-rwxr--r--   3  root  sys     2689  Jan   1  1970  rc6
-rwxr--r--   1  root  sys     9412  Jan   1  1970  rcs
-r-xr-xr-x   2  bin   root  257444  Sep   1  1998  sh
-r-xr-xr-x   1  bin   bin   195300  Sep   1  1998  soconfig
lrwxrwxrwx   1  root  root      13  Nov  27  14:40  su-> ../usr/bin/su
-r-xr-xr-x   1  root  sys   473808  Sep   1  1998  su.static
-r-xr-xr-x   1  root  bin   288544  Sep   1  1998  sulogin
-rwxr--r--   1  root  sys     3138  Jan   1  1970  swapadd
-r-xr-xr-x   1  bin   bin    29736  Sep   1  1998  sync
-r-xr-xr-x   1  root  sys   435004  Sep   1  1998  uadmin
-r-xr-xr-x   1  bin   bin   213408  Sep   1  1998  umount
-r-xr-xr-x   1  root  sys     3292  Jan   1  1970  umountall
-r-xr-xr-x   1  bin   bin   193152  Sep   1  1998  uname
martyp $ un                          ; unescape = 를 입력
1) uname
martyp $ un
```

ksh는 "un"으로 시작하는 지령이 uname뿐이라는것을 결정한다. 지령uname이 렬거되고 "un"은 다시 지령입력재촉문으로 나타난다. 첫 "un" 다음에 입력된 escape=를 볼수 없으므로 그 지령을 보여 주기 위하여 옆에 설명문을 주었다.

"un" 으로 시작되는 모든 파일들을 렬거하기 보다는 아래의 실례에서 보여 주는바와 같이 **un**escape *을 입력하여 현 단어를 모든 파일들로 교체할수 있다.

```
martyp $ cd /sbin
martyp $ ls -al
```

```
total  11704
drwxrwxr-x    2  root   sys       512  Nov  27  14:55  .
drwxr-xr-x   31  root   root     1024  Apr  19  03:24  ..
-r-xr-xr-x    1  bin    bin    200356  Sep   1   1998  autopush
lrwxrwxrwx    1  root   root       21  Nov  27  14:55  bpgetfile -> ../usr/sbin/e
-r-xr-xr-x    1  bin    bin    470436  Sep   1   1998  dhcpagent
-r-xr-xr-x    1  bin    bin    433064  Sep   1   1998  dhcpinfo
-r-xr-xr-x    1  bin    bin    253664  Sep   1   1998  fdisk
-r-xr-xr-x    1  bin    bin    762816  Sep   1   1998  hostconfig
-r-xr-xr-x    1  bin    bin    535900  Sep   1   1998  ifconfig
-r-xr-xr-x    1  root   sys    516484  Sep   1   1998  init
-r-xr-xr-x    2  bin    root   257444  Sep   1   1998  jsh
-r-xr-xr-x    1  bin    bin    224596  Sep   1   1998  mount
-r-xr-xr-x    1  root   sys      6935  Jan   1   1970  mountall
-rwxr--r--    3  root   sys      2689  Jan   1   1970  rc0
-rwxr--r--    1  root   sys      2905  Jan   1   1970  rcl
-rwxr--r--    1  root   sys      2491  Jan   1   1970  rc2
-rwxr--r--    1  root   sys      1948  Jan   1   1970  rc3
-rwxr--r--    3  root   sys      2689  Jan   1   1970  rc5
-rwxr--r--    3  root   sys      2689  Jan   1   1970  rc6
-rwxr--r--    1  root   sys      9412  Jan   1   1970  rcS
-r-xr-xr-x    2  bin    root   257444  Sep   1   1998  sh
-r-xr-xr-x    1  bin    bin    195300  Sep   1   1998  soconfig
lrwxrwxrwx    1  root   root       13  Nov  27  14:40  su ->../usr/bin/su
-r-xr-xr-x    1  root   sys    473808  Sep   1   1998  su.static
-r-xr-xr-x    1  root   bin    288544  Sep   1   1998  sulogin
-rwxr--r--    1  root   sys      3138  Jan   1   1970  swapadd
-r-xr-xr-x    1  bin    bin     29736  Sep   1   1998  sync
-r-xr-xr-x    1  root   sys    435004  Sep   1   1998  uadmin
-r-xr-xr-x    1  bin    bin    213408  Sep   1   1998  umount
-r-xr-xr-x    1  root   sys      3292  Jan   1   1970  umountall
-r-xr-xr-x    1  bin    bin    193152  Sep   1   1998  uname
martyp  $  uname                     ; unescape * 를 입력
```

이 실례에서 uname이 **un**escape *에 맞는 지령에 의하여 입력재촉문의 오른쪽에 교체된다.

다음의 실례는 "um"으로 시작되는 모든 파일들의 목록을 얻기 위하여 umescape=를 리용한것과 마지막으로 현 단어를 "um" 으로 시작되는 첫 파일이름으로 교체하기 위하여 umescape\을 리용한것을 보여 준다.

```
martyp $ ls -al
total 11704
drwxrwxr-x  2  root  sys      512  Nov  27  14:55  .
drwxr-xr-x 31  root  root    1024  Apr  19  03:24  ..
-r-xr-xr-x  1  bin   bin   200356  Sep   1   1998  autopush
lrwxrwxrwx  1  root  root      21  Nov  27  14:55  bpgetfile -> ../usr/sbin/e
-r-xr-xr-x  1  bin   bin   470436  Sep   1   1998  dhcpagent
-r-xr-xr-x  1  bin   bin   433064  Sep   1   1998  dhcpinfo
-r-xr-xr-x  1  bin   bin   253664  Sep   1   1998  fdisk
-r-xr-xr-x  1  bin   bin   762816  Sep   1   1998  hostconfig
-r-xr-xr-x  1  bin   bin   535900  Sep   1   1998  ifconfig
-r-xr-xr-x  1  root  sys   516484  Sep   1   1998  init
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  jsh
-r-xr-xr-x  1  bin   bin   224596  Sep   1   1998  mount
-r-xr-xr-x  1  root  sys     6935  Jan   1   1970  mountall
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc0
-rwxr--r--  1  root  sys     2905  Jan   1   1970  rcl
-rwxr--r--  1  root  sys     2491  Jan   1   1970  rc2
-rwxr--r--  1  root  sys     1948  Jan   1   1970  rc3
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc5
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc6
-rwxr--r--  1  root  sys     9412  Jan   1   1970  rcS
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  sh
-r-xr-xr-x  1  bin   bin   195300  Sep   1   1998  soconfig
lrwxrwxrwx  1  root  root      13  Nov  27  14:40  su ->../usr/bin/su
-r-xr-xr-x  1  root  sys   473808  Sep   1   1998  su.static
-r-xr-xr-x  1  root  bin   288544  Sep   1   1998  sulogin
-rwxr--r--  1  root  sys     3138  Jan   1   1970  swapadd
-r-xr-xr-x  1  bin   bin    29736  Sep   1   1998  sync
-r-xr-xr-x  1  root  sys   435004  Sep   1   1998  uadmin
-r-xr-xr-x  1  bin   bin   213408  Sep   1   1998  umount
-r-xr-xr-x  1  root  sys     3292  Jan   1   1970  umountall
-r-xr-xr-x  1  bin   bin   193152  Sep   1   1998  uname
martyp $ um                        ; unescape = 를 입력
1)  umount
2)  umountall
martyp $ umount  umountall         ; unescape * 를 입력
martyp $ umount                    ; unescape \ 를 입력
```

이제 어느 한 지령과 파일의 이름으로 될수 있는 인수들을 가지고 작업하자. 등록부 /sbin의 "un"처럼 입력된 기호들이 유일하지 않으면 어떤 일이 생기는가? 지령 또는 경로이름이 유일하지 않으면 ksh는 그 지령을 완성하기 위한 선택항목들을 보여 준다.

아래의 실례는 "r"로 시작되는 지령들의 목록을 얻기 위하여 ls **r**escape=를 입력한것을 보여 준다.

```
martyp $ ls -al
total 11704
drwxrwxr-x  2  root  sys      512  Nov  27  14:55  .
drwxr-xr-x 31  root  root    1024  Apr  19  03:24  ..
-r-xr-xr-x  1  bin   bin   200356  Sep   1   1998  autopush
lrwxrwxrwx  1  root  root      21  Nov  27  14:55  bpgetfile -> ../usr/sbin/e
-r-xr-xr-x  1  bin   bin   470436  Sep   1   1998  dhcpagent
-r-xr-xr-x  1  bin   bin   433064  Sep   1   1998  dhcpinfo
-r-xr-xr-x  1  bin   bin   253664  Sep   1   1998  fdisk
-r-xr-xr-x  1  bin   bin   762816  Sep   1   1998  hostconfig
-r-xr-xr-x  1  bin   bin   535900  Sep   1   1998  ifconfig
-r-xr-xr-x  1  root  sys   516484  Sep   1   1998  init
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  jsh
-r-xr-xr-x  1  bin   bin   224596  Sep   1   1998  mount
-r-xr-xr-x  1  root  sys     6935  Jan   1   1970  mountall
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc0
-rwxr--r--  1  root  sys     2905  Jan   1   1970  rcl
-rwxr--r--  1  root  sys     2491  Jan   1   1970  rc2
-rwxr--r--  1  root  sys     1948  Jan   1   1970  rc3
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc5
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc6
-rwxr--r--  1  root  sys     9412  Jan   1   1970  rcS
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  sh
-r-xr-xr-x  1  bin   bin   195300  Sep   1   1998  soconfig
lrwxrwxrwx  1  root  root      13  Nov  27  14:40  su ->../usr/bin/su
-r-xr-xr-x  1  root  sys   473808  Sep   1   1998  su.static
-r-xr-xr-x  1  root  bin   288544  Sep   1   1998  sulogin
-rwxr--r--  1  root  sys     3138  Jan   1   1970  swapadd
-r-xr-xr-x  1  bin   bin    29736  Sep   1   1998  sync
-r-xr-xr-x  1  root  sys   435004  Sep   1   1998  uadmin
-r-xr-xr-x  1  bin   bin   213408  Sep   1   1998  umount
-r-xr-xr-x  1  root  sys     3292  Jan   1   1970  umountall
-r-xr-xr-x  1  bin   bin   193152  Sep   1   1998  uname
```

```
martyp $ ls r                         ; ls  r escape = 를 입력
1)  rc0
2)  rc1
3)  rc2
4)  rc3
5)  rc5
6)  rc6
7)  rcS
martyp $ ls r
```

이 실례로부터 ls -rescape=를 입력하여 7개 파일들의 목록이 생성되였다는것을 알수 있다. 또한 ls r는 다음번 입력재촉문에 배치되였다. 이것을 추가적인 ksh파일이름확장을 수행하기 위한 다음번 입력재촉문에서 리용할수 있다. r로 시작되는 7개 파일들의 목록을 생성하는 ls **r**escape=를 다시 수행하자. 이 입력재촉문은 ls r를 가지며 그 행에 모든 7개파일들의 목록을 생성하는 escape *를 입력한다.

```
martyp $ ls -al
total 11704
drwxrwxr-x  2   root   sys      512   Nov   27   14:55   .
drwxr-xr-x  31  root   root    1024   Apr   19   03:24   ..
-r-xr-xr-x.  1   bin    bin  200356   Sep    1    1998   autopush
lrwxrwxrwx  1   root   root      21   Nov   27   14:55   bpgetfile->../usr/sbin/e
-r-xr-xr-x   1   bin    bin  470436   Sep    1    1998   dhcpagent
-r-xr-xr-x   1   bin    bin  433064   Sep    1    1998   dhcpinfo
-r-xr-xr-x   1   bin    bin  253664   Sep    1    1998   fdisk
-r-xr-xr-x   1   bin    bin  762816   Sep    1    1998   hostconfig
-r-xr-xr-x   1   bin    bin  535900   Sep    1    1998   ifconfig
-r-xr-xr-x   1   root   sys  516484   Sep    1    1998   init
-r-xr-xr-x   2   bin   root  257444   Sep    1    1998   jsh
-r-xr-xr-x   1   bin    bin  224596   Sep    1    1998   mount
-r-xr-xr-x   1   root   sys    6935   Jan    1    1970   mountall
-rwxr--r--   3   root   sys    2689   Jan    1    1970   rc0
-rwxr--r--   1   root   sys    2905   Jan    1    1970   rcl
-rwxr--r--   1   root   sys    2491   Jan    1    1970   rc2
-rwxr--r--   1   root   sys    1948   Jan    1    1970   rc3
-rwxr--r--   3   root   sys    2689   Jan    1    1970   rc5
-rwxr--r--   3   root   sys    2689   Jan    1    1970   rc6
-rwxr--r--   1   root   sys    9412   Jan    1    1970   rcS
-r-xr-xr-x   2   bin   root  257444   Sep    1    1998   sh
```

```
-r-xr-xr-x    1   bin    bin  195300   Sep    1    1998  soconfig
lrwxrwxrwx    1   root   root     13   Nov   27   14:40  su ->../usr/bin/su
-r-xr-xr-x    1   root   sys  473808   Sep    1    1998  su.static
-r-xr-xr-x    1   root   bin  288544   Sep    1    1998  sulogin
-rwxr--r--    1   root   sys    3138   Jan    1    1970  swapadd
-r-xr-xr-x    1   bin    bin   29736   Sep    1    1998  sync
-r-xr-xr-x    1   root   sys  435004   Sep    1    1998  uadmin
-r-xr-xr-x    1   bin    bin  213408   Sep    1    1998  umount
-r-xr-xr-x    1   root   sys    3292   Jan    1    1970  umountall
-r-xr-xr-x    1   bin    bin  193152   Sep    1    1998  uname
```

martyp $ **ls r** ; **ls r** escape = 를 입력

1) rc0
2) rc1
3) rc2
4) rc3
5) rc5
6) rc6
7) rcS

martyp $ **ls rc0 rc1 rc2 rc3 rc5 rc6 rcS** ;7개 파일들을 얻기 위해 escape*을 입력

escape *은 r로 시작되는 7개 파일들을 렬거하고 그것들을 지령행에 배치한다.

다음 실레에서는 escape=와 escape *을 주고 그다음 ls escape_을 입력하여 마지막지령의 마지막단어 rcS를 지령행에 가져 온다.

martyp $ **ls -al**

```
total 11704
drwxrwxr-x    2   root   sys     512   Nov   27   14:55  .
drwxr-xr-x   31   root   root   1024   Apr   19   03:24  ..
-r-xr-xr-x    1   bin    bin  200356   Sep    1    1998  autopush
lrwxrwxrwx    1   root   root     21   Nov   27   14:55  bpgetfile -> ../usr/sbin/e
-r-xr-xr-x    1   bin    bin  470436   Sep    1    1998  dhcpagent
-r-xr-xr-x    1   bin    bin  433064   Sep    1    1998  dhcpinfo
-r-xr-xr-x    1   bin    bin  253664   Sep    1    1998  fdisk
-r-xr-xr-x    1   bin    bin  762816   Sep    1    1998  hostconfig
-r-xr-xr-x    1   bin    bin  535900   Sep    1    1998  ifconfig
-r-xr-xr-x    1   root   sys  516484   Sep    1    1998  init
-r-xr-xr-x    2   bin    root 257444   Sep    1    1998  jsh
-r-xr-xr-x    1   bin    bin  224596   Sep    1    1998  mount
```

```
-r-xr-xr-x    1  root   sys    6935  Jan   1  1970  mountall
-rwxr--r--    3  root   sys    2689  Jan   1  1970  rc0
-rwxr--r--    1  root   sys    2905  Jan   1  1970  rcl
-rwxr--r--    1  root   sys    2491  Jan   1  1970  rc2
-rwxr--r--    1  root   sys    1948  Jan   1  1970  rc3
-rwxr--r--    3  root   sys    2689  Jan   1  1970  rc5
-rwxr--r--    3  root   sys    2689  Jan   1  1970  rc6
-rwxr--r--    1  root   sys    9412  Jan   1  1970  rcS
-r-xr-xr-x    2   bin  root  257444  Sep   1  1998  sh
-r-xr-xr-x    1   bin   bin  195300  Sep   1  1998  soconfig
lrwxrwxrwx    1  root  root      13  Nov  27 14:40  su ->../usr/bin/su
-r-xr-xr-x    1  root   sys  473808  Sep   1  1998  su.static
-r-xr-xr-x    1  root   bin  288544  Sep   1  1998  sulogin
-rwxr--r--    1  root   sys    3138  Jan   1  1970  swapadd
-r-xr-xr-x    1   bin   bin   29736  Sep   1  1998  sync
-r-xr-xr-x    1  root   sys  435004  Sep   1  1998  uadmin
-r-xr-xr-x    1   bin   bin  213408  Sep   1  1998  umount
-r-xr-xr-x    1  root   sys    3292  Jan   1  1970  umountall
-r-xr-xr-x    1   bin   bin  193152  Sep   1  1998  uname
martyp $ ls r                          ; ls  r escape = 를 입력
1)  rc0
2)  rc1
3)  rc2
4)  rc3
5)  rc5
6)  rc6
7)  rcS
martyp  $  ls  rc0  rc1  rc2  rc3  rc5  rc6  rcS; 7개 파일들을 얻기 위해
                                         escape * 를 입력
rc0   rc1   rc2   rc3   rc5  rc6   rcS
martyp  $  ls  rcS                     ; 마지막단어를 얻기 위해
                                         ls escape_을 입력
rcS
martyp  $
```

escape3_을 입력하여 세번째 단어를 규정해 줄수 있다.
아래의 실례에서는 마지막지령의 세번째 단어를 얻는것을 보여 주고 있다.

```
martyp $ ls -al
total 11704
drwxrwxr-x  2  root  sys      512  Nov  27  14:55  .
drwxr-xr-x 31  root  root    1024  Apr  19  03:24  ..
-r-xr-xr-x  1  bin   bin   200356  Sep   1   1998  autopush
lrwxrwxrwx  1  root  root      21  Nov  27  14:55  bpgetfile -> ../usr/sbin/e
-r-xr-xr-x  1  bin   bin   470436  Sep   1   1998  dhcpagent
-r-xr-xr-x  1  bin   bin   433064  Sep   1   1998  dhcpinfo
-r-xr-xr-x  1  bin   bin   253664  Sep   1   1998  fdisk
-r-xr-xr-x  1  bin   bin   762816  Sep   1   1998  hostconfig
-r-xr-xr-x  1  bin   bin   535900  Sep   1   1998  ifconfig
-r-xr-xr-x  1  root  sys   516484  Sep   1   1998  init
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  jsh
-r-xr-xr-x  1  bin   bin   224596  Sep   1   1998  mount
-r-xr-xr-x  1  root  sys     6935  Jan   1   1970  mountall
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc0
-rwxr--r--  1  root  sys     2905  Jan   1   1970  rcl
-rwxr--r--  1  root  sys     2491  Jan   1   1970  rc2
-rwxr--r--  1  root  sys     1948  Jan   1   1970  rc3
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc5
-rwxr--r--  3  root  sys     2689  Jan   1   1970  rc6
-rwxr--r--  1  root  sys     9412  Jan   1   1970  rcS
-r-xr-xr-x  2  bin   root  257444  Sep   1   1998  sh
-r-xr-xr-x  1  bin   bin   195300  Sep   1   1998  soconfig
lrwxrwxrwx  1  root  root      13  Nov  27  14:40  su ->../usr/bin/su
-r-xr-xr-x  1  root  sys   473808  Sep   1   1998  su.static
-r-xr-xr-x  1  root  bin   288544  Sep   1   1998  sulogin
-rwxr--r--  1  root  sys     3138  Jan   1   1970  swapadd
-r-xr-xr-x  1  bin   bin    29736  Sep   1   1998  sync
-r-xr-xr-x  1  root  sys   435004  Sep   1   1998  uadmin
-r-xr-xr-x  1  bin   bin   213408  Sep   1   1998  umount
-r-xr-xr-x  1  root  sys     3292  Jan   1   1970  umountall
-r-xr-xr-x  1  bin   bin   193152  Sep   1   1998  uname
martyp $ ls r                                  ; ls  r escape = 를 입력
1)  rc0
2)  rc1
3)  rc2
4)  rc3
5)  rc5
6)  rc6
```

7) rcS

martyp $ **ls rc0 rc1 rc2 rc3 rc5 rc6 rcS** ; 7개 파일들을 얻기 위해
escape * 를 입력

rc0 rc1 rc2 rc3 rc5 rc6 rcS ; 세번째 단어를 얻기 위해

martyp $ **ls rc1** **ls** escape**3**_을 입력

앞의 지령에는 ls지령이 포함되여 있으며 그것이 첫번째 단어로 되기때문에 세번째 단어는 rc3이 아니라 rcl이다.

표 10-1에서는 앞의 실례들에서 리용된 지령과 경로의 완성방법을 보여 주고 있다.

표 10-1 ksh에서 지령과 경로의 완성

단어 또는 지령 (매번 먼저 escape를 입력)	설 명
wordescape=	word로 시작되는 파일이름들의 번호 붙은 목록을 현시
wordescape*	word로 맞는 모든 파일들로 교체
wordescape\	word를 word로 시작하는 첫 파일이름으로 교체
command **word**escape=	word로 시작하는 파일이름들의 번호 붙은 목록을 현시
command **word**escape*	word로 맞는 파일들로 교체
command **word**escape_	마지막지령의 마지막단어를 유표위치에 삽입
command **word**escape_3	마지막지령의 세번째 단어를 유표위치에 삽입

파일이름확장

ksh에서 작업할 때 흔히 파일이름과 관련되는 일들을 적지 않게 수행한다. 파일이름들을 취급하는 쉘스크립트들을 작성하기전에 파일이름확장에 대하여 알아 두는것이 좋다.

표10-2는 몇가지 공통적인 파일이름확장과 패턴정합을 보여 준다.

표 10-2에 있는 실례들을 자세히 서술하면 다음과 같다.

① 등록부에서 ".c"로 끝나는 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls *.c
  conf.SAM.c  conf.c
```

② 등록부에서 "conf"로 이름 지어 져 있고 한개 기호로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls conf.?
  conf.c   conf.o  conf.1
```

표 10-2 파일이름확장과 패턴정합

기호들	실 례	설 명
*	1) ls*. c	령 또는 그이상 기호들을 맞춘다.
?	2) ls conf. ?	임의의 한개 기호를 맞춘다.
[list]	3) ls conf. [co]	목록에 있는 임의의 기호를 맞춘다.
[lower-upper]	4) ls libdd. 9873[5-6]. sl	범위에 있는 임의의 기호를 맞춘다.
str{str1, str2, str3, …}	5) ls ux*. {700, 300}	str를 { }의 내용으로 확장한다.
~	6) ls -a~	홈등록부
~username	7) ls -a ~gene	username의 홈등록부

③ 등록부에서 "conf"로 이름 지어 져 있고 한개 기호 "c" 또는 "o"로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls conf. {co}
  conf. c   conf. o
```

④ 류사한 이름들을 가지면서 어떤 범위의 마당들을 가지는 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls libdd9873[5-6]. sl
  libdd98735. sl libdd98736. sl
```

⑤ "ux"로 시작되고 확장자가 "300" 또는 "700"인 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls ux*. {700, 300}
  uxbootlf. 700 uxinstfs. 300
```

⑥ 홈등록부에 있는 파일들을 렬거하려면 아래에서 보여 주는것처럼 "～"를 리용한다.

```
$ ls -a~
  .          .cshrc.org   .login       .shrc.org
  ..         .exrc        .login.org   .cshrc
  .history
```

⑦ 사용자의 홈등록부에 있는 파일들을 렬거하려면 다음과 같이 한다.

```
$ ls -a -gene
```

```
.            .history       splinedat      under.des
..           .login         trail.txt      xtra.part
.chsrc       .login.org     ESP-File
.cshrc.org   .profile       Mail
.exrc        .shrc.org      opt
```

이 기술들은 ksh에서 작업할 때와 쉘스크립트를 작성할 때 리용되므로 파일이름확장에 익숙되여야 한다.

방향바꾸기(I/O방향바꾸기)

UNIX는 지령들이 입력을 건반(표준입구장치)으로부터 받으며 출력을 화면(표준출구장치)에 보내도록 설치되여 있다. 지령들은 오유정보도 화면에 보낸다. 이 표준설정을 무시하고 입력이 다른 곳으로부터 들어 오고 출력과 오유들이 다른 곳으로 나가게 할수도 있다. 이것을 방향바꾸기라고 부른다. 표10-3는 방향바꾸기의 형식들을 보여 준다.

표에서 보는바와 같이 지령의 출력을 표준출구장치로부터 파일에로 전환하려면 ">"을 리용하여야 한다. noclobber라고 부르는 환경변수를 설정하면 이미 존재하는 파일에로의 방향바꾸기는 진행되지 않는다. 실례로 다음의 /tmp/processes와 같은 이미 존재하는 파일에 기입하려고 시도한다면 그 파일이 존재한다는 통보문을 받는다.

```
# ps -ef > /tmp/processes
/tmp/processes: File exists
```

그러나 파일을 덧 쓰도록 방향바꾸기에 "!"를 리용할수 있다. ">!"를 리용하면 파일은 덧 씌여 진다. 그리고 ">>!"은 이미 존재하는 파일의 뒤에 출력이 첨가되도록 한다. 이에 대한 실례들을 표 10-3에서 보여 준다.

표 10-3 흔히 리용되는 방향바꾸기형식들

지령 또는 대입	실 례	설 명
<	wc -l < .login	표준입력방향바꾸기 : wc(단어계수)를 실행하고 .login에 있는 행들의 수를 렬거한다.
>	ps -ef > /tmp/processes	표준출력방향바꾸기 : ps를 실행하고 출력을 파일 /tmp/process에로 보낸다.
>>	ps -ef >> /tmp/processes	표준출력을 첨가 : ps를 실행하고 출력을 파일 /tmp/process 에 첨가.

(표계속)

지령 또는 대입	실 례	설 명
>!	ps -ef > ! /tmp/processes	출력방향바꾸기를 첨가하고 noclobber를 무시한다. /tmp/processes가 존재하면 그우에 덧쓴다.
>>!	ps -ef >>! /tmp/processes	표준출력을 첨가하고 noclobber를 무시한다. /tmp/processes의 끝에 첨가한다.
| (pipe)	ps | wc –l	ps를 실행하고 그 결과를 wc의 입력으로 리용한다.
0-표준입력		
1-표준출력		
2-표준오유	cat program 2> errors	파일program을 표준출력에 cat하고 오유들을 파일 errors에로 방향바꾸기한다.
	cat program 2>> errors	파일program을 표준출력에 cat하고 오유들을 파일 errors에로 첨가한다.
	find / -name '*.c' –print >cprograms 2>errors	체계상에서 .c로 끝나는 한 파일들을 찾고 파일들의 목록을 현 작업등록부에 있는 cprograms에 배치한다. 그리고 모든 오유들을 (파일서술자2) 현 작업등록부에 있는 파일 errors에 보낸다.
	flnd / -name '*.c' –print >cprograms 2>&1	체계상에서 .c로 끝나는 모든 파일들을 찾고 파일목록을 현작업등록부에 있는 cprograms에 배치한다. 그리고 모든 오유(파일서술자 2)들을 파일서술자 1(cprograms)이 있는 동일한 위치에로 보낸다.

환경변수

환경변수는 기호렬에 결합된 이름이다. 이름은 변수이고 기호렬은 값이다. 환경변수들은 부분쉘들에서 또는 쉘에 의하여 파생된 프로쎄스들에서 리용될수 있다. 대부분의 체계들에서 환경변수들은 공통적으로 대문자로 표시된다.

체계상에서 지령을 줄 때 보통 절대적인 경로이름이 아니라 상대적인 경로이름을 입력한다. 변수 PATH는 지령들이 배치되여 있는 등록부의 위치를 가리킨다. 이 변수가 없으면 매 지령의 절대적인 경로이름을 지적해 주어야 한다. 지령을 줄 때 쉘은 변수 PATH에 렬거된 등록부들을 순서대로 탐색한다. 설정된 환경변수들을 보기 위한 좋은 방법은 아래에서 보여 주는바와 같이 지령 env를 리용하는것이다.

```
martyp $ env
_=/usr/bin/env
MANPATH=:/opt/local/ptc/sysadmin/scripts:/opt/local/ptc/man:/opt/local/altrasofn
_INIT_UTS RELEASE=5.7
HZ=100
_INIT_UTS_MACHINE=sun4m
EPC=true
PATH=/usr/bin:/usr/ucb:/etc: .
WEB_SERVER=sioux.rose.hp.com
_INIT_UTS_VERSION=Generic
MODEL=SPARCstation-10
OS_REV=5 : 7
EDITOR=vi
_INIT_RUN_NPREV=O
CLASSPATH=.:/usr/java/lib:
LOGNAME=martyp
_INIT_UTS_NODENAME=sunsys
_INIT_UTS_ISA=sparc
MAIL=/var/mail/martyp
ERASE=^H
OS=solaris
PS1=$PWD
$LOGNAME $TOKEN
_INIT_PREV_LEVEL=S
HOST=sunsys
TESTEXPERT_HOME=/opt/local/svn/te33
TA_HOME=/opt/local/platinum/solaris2/testadvise
MA_HOME=/opt/local/platinum/sclaris2/memadvise
CL_LICENSE FILE=/opt/local/CenterLine/configs/license.dat
SHELL=/bin-/ksh
PROFILE DIR=/opt/local/ptc/sysadmin/profile.d
OSTYPE=solaris2
HOME=/home/martyp
```

```
_INIT_UTS_SYSNAME=SunOS
_TERM=vt100
LD_LIBRARY_PATH= : /opt/local/parasoft/lib. solaris
MWHOME=/op-E/local/mainsoft/mainwin/mw
FMHOME=/opt/local/adobe/frame
PWD=/home/martyp
TZ=US/Pacific
_INIT_RUN_LEVEL=3
CLEARCASE_BLD_UMASK=02
_INIT_UTS_PLATFORM=SUNW, SPARCstation-10
martyp $
```

여기서 볼수 있는바와 같이 PATH외에도 다른 환경변수들이 UNIX체계상에서의 작업을 쉽게 해준다. ksh 변수들과 관련된 다른 지령들을 줄수 있다. 표 10-4에서는 ksh 변수들과 관련된 지령들을 보여 주고 있다.

표 10-4 ksh관련의 변수지령들

지 령	설 명
Env	모든 환경변수 (반출된)들을 렬거한다. 이 변수들은 보통 대문자로 되여 있으며 후손프로쎄스에 전달된다.
Set	모든 국부적변수 및 반출된 변수들을 인쇄한다.
set-0	on 또는 off로 설정된 모든 내장변수들을 렬거한다.
typeset	모든 변수들과 결합된 속성들, 함수들, 옹근수들을 현시한다.
typeset +	변수들의 이름들만을 현시한다.

어떤 환경변수들이 설정되여 있는가를 보기 위해서는 이 모든 지령들을 체계상에서 실행시켜 볼수 있다. 특정한 환경변수들의 값을 알려면 echo지령을 리용해야 한다. 아래의 실례에서는 환경변수 HOME의 값을 보기 위하여 echo지령을 주는것을 보여 주고 있다.

martyp $ **echo $HOME**
/home/martyp

류사하게 특정한 콤퓨터상의 조작체계형을 보려면 다음의 지령을 줄수 있다.

martyp $ **echo $OSTYPE**
solaris2

환경변수이름의 앞에 붙은 $는 그 변수의 값이 표준출력으로 보내진다는것을 규정한다. 이 경우에 환경변수 HOME의 값은 /home/martyp이다. 이것은 현 사용자의 홈등록부

가 /home/martyp이라는것을 의미한다. 다음의 형식으로 ksh에서 사용자자신의 환경변수들을 정의할수 있다.

```
export   NAME=value
```

많은 사용자들은 ksh의 입력재촉문을 자기의 요구에 맞게 정할수 있다. 대부분의 체계들은 표준적으로 4개의 ksh입력재촉문을 제공한다. 첫 두개의 입력재촉문은 변경시킬수 있다. 첫번째 입력재촉문은 1차입력재촉문인 PS1이다. 두번째 입력재촉문 PS2는 지령을 부분적으로 입력하고 enter를 눌렀을 때 나타난다. PS1은 표준적으로 $에, PS2은 >에 설정되여 있다. 아래의 실례에서는 PS1을 홈등록부, 공백, 현재 리력번호, 공백, 반점, 기호 $로 설정한다.

PS1=" `pwd` ! $ "

PS2에 대하여서는 표준값 >를 그대로 놓는다. 아래에서는 새로운 PS1입력재촉문과 기정PS2입력재촉문에 지령을 주는것을 보여 주고 있다.

/home /martyp 187 $ **print " This is new PS1>but default PS2"**
This is new PS1
but default PS2
/home /martyp 188 $

이 실례는 187로부터 188에로 증가한 리력항목을 포함하는 새로운 PS1과 완성되지 않은 print지령을 주었을 때의 기정 PS2를 보여 준다.

이미 존재하는 변수의 끝에 다음과 같은 형식으로 첨가할수도 있다.

```
export NAME="$NAME: appended_information"
```

실례로 /home/martyp/programs를 이미 존재하는 환경변수에 첨가하려면 다음의 지령을 준다.

```
export PATH="$PATH:/home/martyp/programs"
```

이것은 경로 /home/martyp/programs를 환경변수 PATH에 첨가한다.

ksh와의 작업은 매우 유연하다. 사용자는 자기의 일감을 쉽게 하는 모든 변수들을 볼수 있으며 변경시킬수 있다. 사용자는 PS1과 PATH를 자기의 요구에 맞게 변경시켜 일감을 쉽게 처리할수 있다.

배경일감과 일감조종

지금까지 본 많은 실례들에서처럼 지령을 실행시키면 그 지령이 완성될 때까지 입력재촉문이 나타나지 않는다. 일부 지령들은 완성되는데 많은 시간을 소비하므로 입력재촉문이 다시 나타날 때까지 오래동안 기다려야 한다. 입력재촉문이 다시 나타나기를 기다리면서 그 지령을 배경에서 실행시킬수 있다. UNIX는 다중과제체계이기때문에 많은 지령들을 배경에서 실행시키고 다른 지령들을 더 줄수 있도록 입력재촉문을 제공해 준다.

지령을 배경에서 실행시키려면 간단히 지령행의 끝에 &를 첨가하여야 한다. 지령의 끝에 한개의 &를 붙이면 그 지령은 배경에서 실행되며 입력재촉문은 곧 다시 나타난다.

이제 몇가지 지령들을 Solaris체계에서 실행시켜 보자. 우선 ".c"로 끝나며 완결되는데 일정한 시간이 걸리는 모든 파일들을 /usr에서 찾기 위한 지령을 실행시키자. 기호렬 find를 time지령과 함께 리용하여 그 지령이 완결되는데 얼마나 오랜 시간이 걸리는가를 보기로 하자.

```
martyp $ time find /usr -name *.c
          ⁞
/usr/demo/link_audit/src/dumpbind.c
/usr/demo/link_audit/src/env.c
/usr/demo/link_audit/src/hash.c
/usr/demo/link_audit/src/perfcnt.c
/usr/demo/link_audit/src/symbindrep.c
/usr/demo/link_audit/src/truss.c
/usr/demo/link_audit/src/who.c
find: cannot read dir /usr/aset: Permission denied

real      1m21.04s
user      0ml.51s
sys       0m12.67s
```

이 지령이 완결되는데 대략 1min 21s 걸렸다. 이 지령이 전경에서 실행되였기때문에 그동안 다른 지령을 줄수 없었다. 왜냐하면 입력재촉문이 되돌려 질 때까지 기다려야 했기때문이다.

지령을 배경에서 실행시키고 입력재촉문을 즉시 되돌려 받기 위해서는 아래의 실례에서 보여 주는바와 같이 지령을 줄 때 뒤에 &를 붙여 주면 된다.

```
martyp $ time find /usr -name *.c > cprogs 2>&1 &
[31 16279
martyp $
real      2m10.20s
user      0ml.31s
sys       0m8.62s
```

이 지령을 배경에서 실행시킨 결과는 첫째로 각괄호안에 넣은 일감번호이며 둘째로 프로쎄스id(PID)이다. 오유들을 포함하여 이 지령의 모든 출력들은 파일 cprogs에 기입된다. 입력재촉문은 지령을 준 다음 즉시 되돌려 지며 그 지령이 완결된 후에 time의 출력이 화면에 현시된다. 배경에서 실행되도록 지령을 준 다음 즉시 다른 지령을 주기

시작할수 있다. 사용자는 전경일감과 배경일감들을 조종할수 있다. 전경일감을 정지시키려면 아래의 실례에서 보여 준바와 같이 ^z(crtl-z)를 누르면 된다.

```
martyp $ find /usr -name *.c
/usr/openwin/share/include/X11/Xaw/Template.c
/usr/openwin/share/src/dig_samples/DnD/main.c
/usr/openwin/share/src/dig_samples/DnD/owner.c
/usr/openwin/share/src/dig_samples/DnD/requestor.c
/usr/openwin/share/src/dig_samples/Tooltalk/olit_tt.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_callbacks.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_code.c
/usr/openwin/share/src/dig_samples/cel/ce-map1.c
/usr/openwin/share/src/dig_samples/ce2/ce_simple.c
/usr/openwin/share/src/dig_samples/dnd_olit/olitdnd.c
/usr/openwin/share/src/dig_samples/dnd_xviewl/xview_dnd.c
/usr/openwin/share/src/dig_samples/dnd-xview2/xview_dnd2.c
/usr/openwin/share/src/dig_samples/selection_olit/olit_sel.c
/usr/openwin/share/src/dig-samples/tooltalk_simple/tt-send.c
/usr/openwin/share/src/olit/oldials/oldials.c
/usr/openwin/share/src/olit/olitbook/ch10/draw.c
^Z[3] + Stopped (SIGTSTP)        find /usr -name *.c
```

ctrl-z를 누르면 find지령은 정지되며 이때 사용자는 일감번호(이 경우에 3)를 볼수 있고 그 일감의 상태는 "Stopped"로 표시된다. 이 지령은 완전히 끝난것이 아니라 일시정지되였을뿐이다. 이 프로쎄스를 지령 fg로 전경에서 실행시키거나 지령 bg로 배경에서 실행시킬수 있다. 지령 bg는 지령의 뒤에 &를 붙여 준것과 같다. 그 지령은 중단된 위치에서 시작된다. 지령 fg 또는 bg을 줄 때 일감번호를 지적하면 안된다. 왜냐하면 표준적으로 마지막일감(이 경우에 일감번호 3)에서 규정된 조작을 수행하기때문이다.

이 실례에서는 일감번호 3을 정지시켰다. 이것은 보다 작은 번호를 가지고 실행되는 다른 일감들이 있다는것을 의미한다. 지령 jobs를 리용하여 모든 일감들의 목록과 그것들의 상태를 얻을수 있다. 또한 지령이 전경에서 실행되도록 fg에 %와 일감번호를 붙여 주거나 지령이 배경에서 실행되도록 bg에 %와 일감번호를 붙여 주는 방법으로 일감들을 조종할수 있다. 일감을 종결시키려면 kill지령에 %와 일감번호를 붙여 주면 된다.

아래의 실례에서는 지령 jobs로 모든 지령들을 렬거하는것과 kill지령으로 일감 1과 일감 3를 제거하며 일감 3을 배경에서 실행시키는것을 보여 주고 있다.

```
martyp $ jobs
[3] + Stopped (SIGTSTP) find /usr -name *.c
[2] - Running        time find / -name gnu* > gnu 2>&l &
[1]   Running        time find / -name *.c > cprogs2>&:l &
```

```
martyp $ kill %l
[1] Terminated      time find / -name *.c > cprogs2>&l &
martyp $ kill %2
[2] - Terminated   time find / -name gnu* >.gnu 2>&l &
martyp $ bg %3
[3]   find /usr -name *.c&
martyp $
```

일감 3이 배경에서 실행되도록 &를 붙여 주었다. 일감 1과 일감 2를 제거하고 일감 3을 배경에서 실행시키면 입력재촉문을 즉시 돌려 주므로 사용자는 다른 일을 수행할수 있다.

umask와 허락

ksh와 관련하여 취급할 추가적인 문제는 파일허락들과 그것들이 umask에 관련되는 방식이다. 이것은 모든 사람들이 리용할수 있는 쉘프로그람들과 제한된 수의 사용자들(가능하게는 체계관리자)만이 리용할수 있는 쉘프로그람들을 작성하려고 하는 경우에 중요한 문제로 제기된다. umask는 새 파일들과 등록부들에 대한 허락을 규정하는데 리용된다.

다음의 실례를 고찰하자. 이 실례에서는 ls -l의 별명 ll을 리용하고 있다.

```
sys1 1: ls  -l  script1
-rwxr-xr-x 1 marty users 120 Jul 26 10:20 script1
```

이 파일에 대한 접근허락은 지령 ll을 줄 때 읽기(r), 쓰기(w), 실행(x)의 위치에 의하여 정의된다. 그림 10-1는 이 파일에 대한 3가지 접근허락들의 그룹들을 보여 준다.

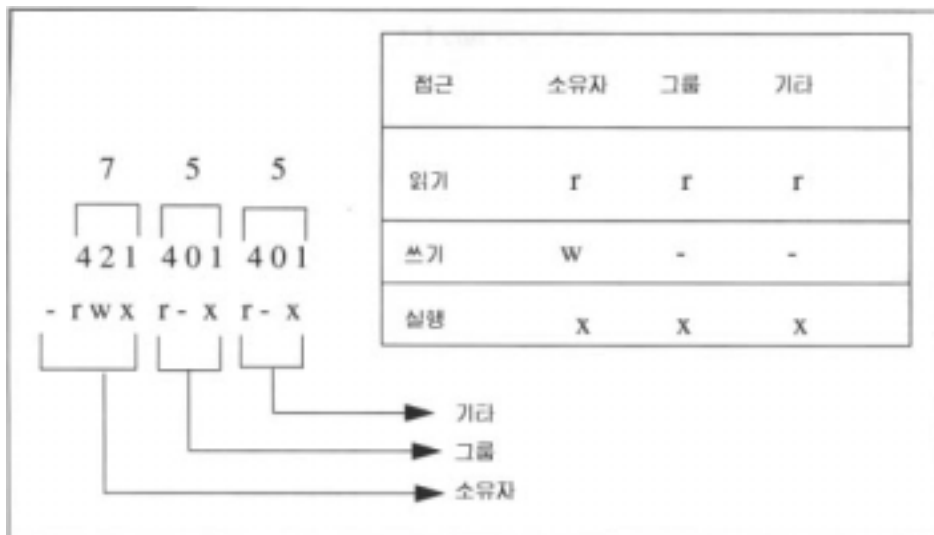



그림 10-1. 파일허락의 실례

파일의 소유자는 자기의 파일에 대한 읽기, 쓰기, 실행허락을 가진다. 사용자가 속하는 그룹과 기타 다른 사람들은 읽기와 실행허락을 가진다. 파일에 대한 허락들은 매 마당의 8진합에 의해 규정된다. 이 경우에는 755이다.

새로운 쉘스크립트 또는 임의의 새로운 파일을 만들면 어떤 문제가 제기되는가, 어떤 허락설정들이 존재하는가 하는 문제가 제기된다. 쉘스크립트를 실행시키려면 이 파일에 대한 실행허락이 필요된다. umask를 리용하여 모든 새로운 파일들과 등록부들에 대한 기정값을 정의할수 있다.

기정적으로 대부분의 체계들은 등록부에서 대해서는 777, 파일들에 대해서는 666의 허락을 가지고 시작한다. 이것은 누구나 다 모든 등록부들에 대해서는 완전한 접근을 가지며 파일들에 대해서는 읽기와 쓰기접근을 가진다는것을 의미한다. 이 표준값들은 umask의 값으로 변경된다.

다음과 같은 두가지 방법으로 umask를 볼수 있다.

```
martyp $ umask
002
martyp $ umask -S
u=rwx, g=rwx, o=rx
martyp $
```

첫번째 실례는 umask의 8진값을 현시하며 두번째 실례는 umask의 기호적인 값을 보여 준다.

umask는 접근을 불가능하게 하며 3개의 8진마당들을 리용한다. 그 마당들은 그림 10-2에서 보여 주는바와 같이 사용자와 그룹들 그리고 기타 다른 사람들에 대한 접근코드들의 합이다.

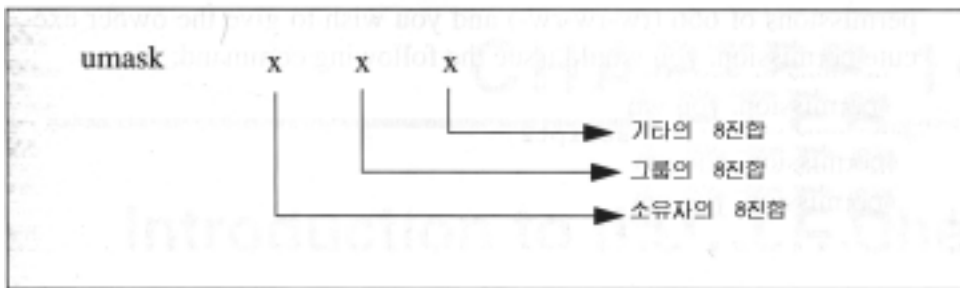



그림 10-2. umask의 마당들

umask마당의 부정은 표준설정과 론리적되여 umask를 변경시킨다. umask의 값을 설정해 줄수 있다. 앞의 실례에서는 umask를 두가지 방법으로 보았다. umask를 설정하려면 간단히 umask지령과 요구되는 값을 주면 된다. 실례로 umask를 022에 설정하면 그림 10-3에서 보여 주는것처럼 "group"과 "other"에 대하여 등록부의 쓰기허락이 제거된다.

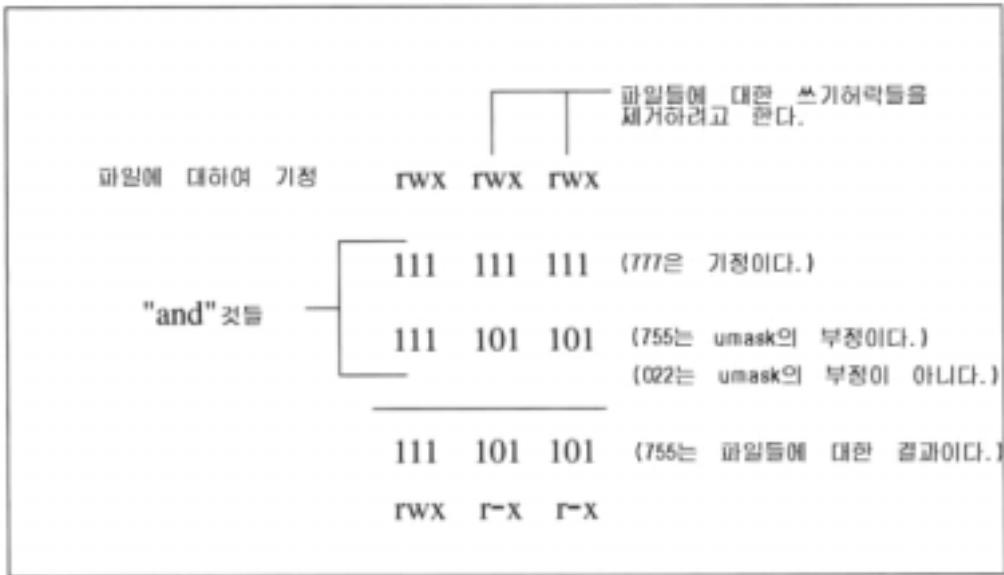



그림 10-3. umask실례

이 실례에서 umask 022는 파일허락을 755로 변경시킨다.

류사하게 022인 umask는 파일들에 대하여 666인 허락을 664로 변경시킨다. 이것은 그룹과 기타에 대하여 읽기허락만을 의미한다.

chmod로 파일허락의 변경

chmod지령은 파일에 대한 허락을 변경시키는데 리용된다. 우에서 서술된바와 같이 umask로 무엇을 할수 있는가에는 상관없이 chmod로 임의의 시각에 파일들에 대한 허락을 변경시킬수 있다. 대부분의 경우들에 chmod로 파일허락을 변경시키는것은 그 파일의 소유자 또는 **상급사용자**(superuser)만이 할수 있다. chmod에 대한 론의를 sort를 가지고 시작하자.

```
$ ls -l sort
-rwxr-x--x    1 marty        users     120  Jul  26  10:20  sort
```

그림 10-4에서는 sort에 대한 허락의 분해를 보여 주고 있다.

사용자들은 정의되는 파일형에 대하여 조종을 많이 할수 없다. 그러나 파일이 사용자의것일 때에는 그 파일의 허락을 마음대로 조종할수 있다. chmod지령은 파일 또는 등록부에 대한 허락을 변경시키는데 리용된다. 만일 사용자가 그 파일의 소유자이면 사용자는 그 파일에 대한 허락을 변경하는 마당만을 가질수 있다.

허락을 변경시킬수 있는 방법은 두가지 즉 기호적인것과 수값적인것이 있다. 수들을 관리하는것이 더 쉬우므로 우선 수값방식을 취급하고 그다음에 기호들을 취급하며 chmod개요에 기호들의 의미들을 포함시킨다.

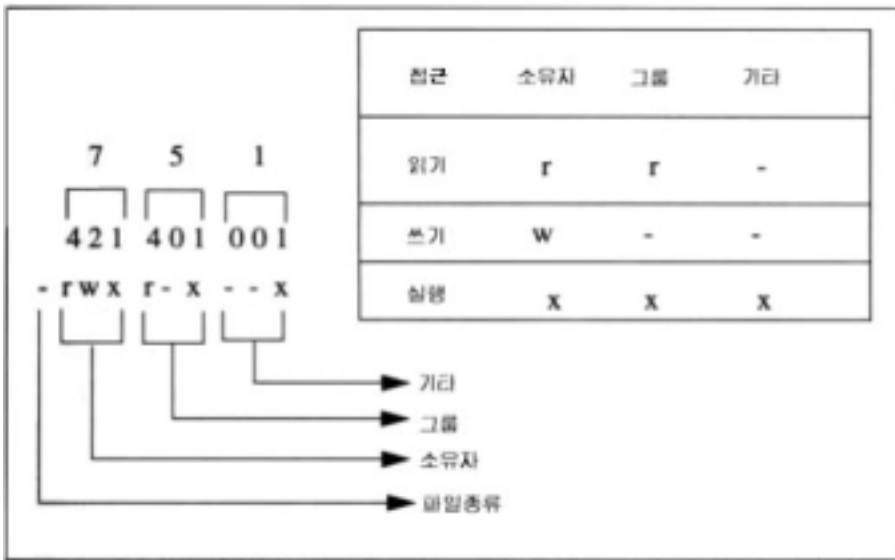



그림 10−4. 파일 sort의 허락들

수라는것은 무엇을 의미하는가? sort에 대한 수들을 보면 751이라는 허락이 있다. 7은 소유자(백의 자리), 5는 그룹(열의 자리), 1은 기타(일의 자리)에 대한것이다. 그림 10−5에서는 위치들의 의미를 설명해 주고 있다.

owner, group, other에 대하여 요구되는 허락을 선택하려면 chmod지령을 리용하여 그 허락들을 파일 또는 등록부에 할당한다. 이 허락가능성들중의 일부(실례로 실행허락)는 그리 리용되지 않는다. 왜냐하면 파일을 실행시키기 위해서는 그 파일이 읽기허락으로 되여 있어야 하기때문이다. 그러나 그림 10−5에서는 모든 가능성들을 다 포함시켰다. 그림 10−5에서 보여 준 허락방식비트들외에도 현재는 필요되지 않는 기타 방식비트들도 있다.

만일 group에 대하여 sort의 쓰기허락을 첨가하고 other에 대하여 모든 허락들을 제거하려면 chmod지령을 줄 때 알맞는 수값을 함께 주면 된다. 아래의 지령모임은 우선 sort에 대하여 이미 존재하는 허락들을 렬거하고 변경시키며 마지막으로 새로운 허락을 렬거한다.

```
$ ls -l sort
-rwxr-x--x       1 marty users  120 Jul 26 10:20 sort

$ chmod 770 sort

$ ls -l sort
-rwxrwx ---      1 marty users  120 Jul 26 10:20 sort
```

기호방식을 리용하여 허락들을 변경시키는 동일한 지령모임은 실례로 다음과 같다.

```
$ ls -l sort
-rwxr-x--x       1 marty users  120 Jul 26 10:20 sort
```

```
$ chmod g+w, o-x sort

$ ls -l sort
-rwxrwx ---          1 marty users   120 Jul 26 10:20 sort
```

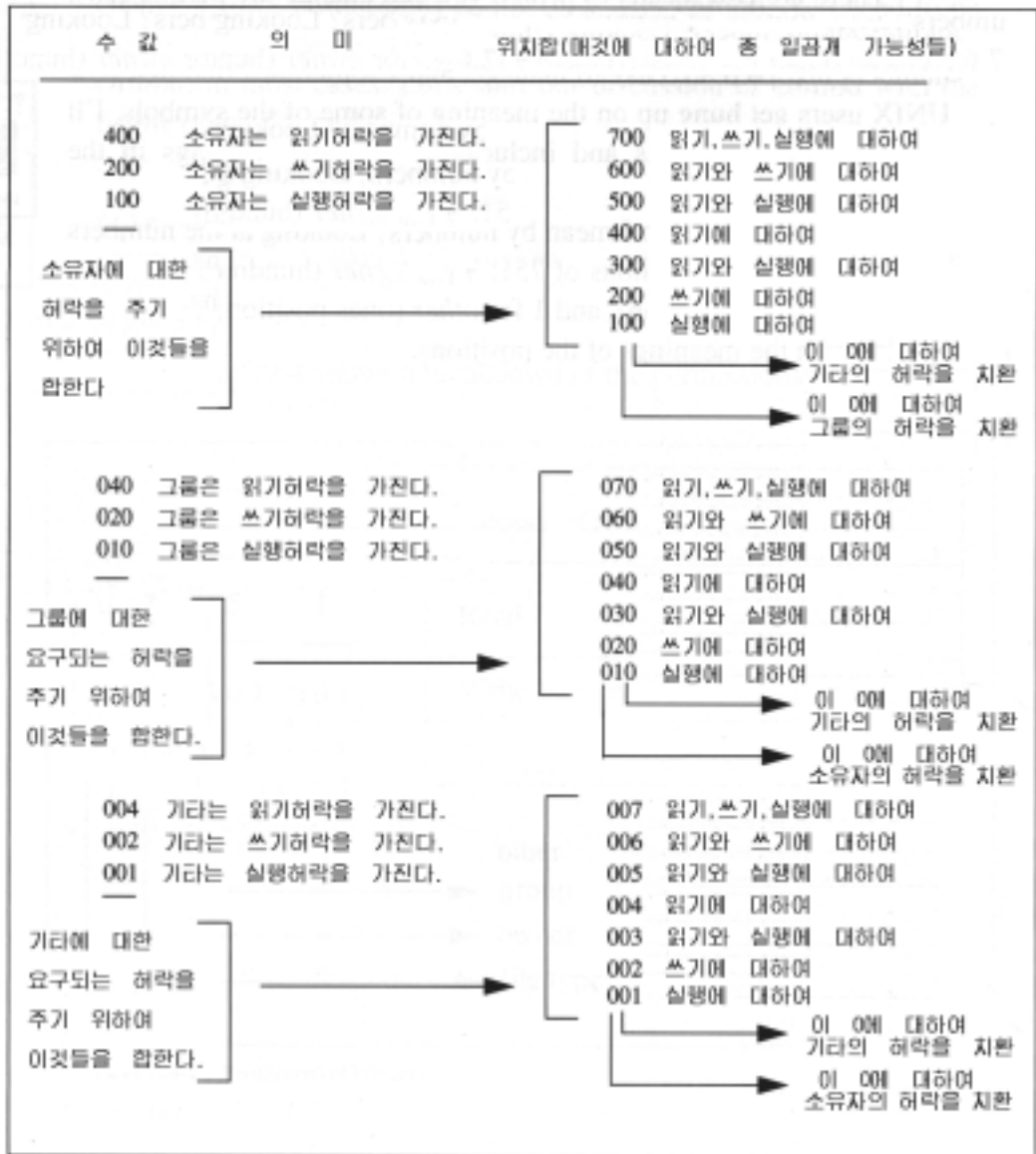



그림 10-5. 수값허락들의 개괄

기호방식에서는 chmod지령을 주고 변경에 의해서 영향을 받을 사람(사용자(u), 그룹(g), 기타(o), 모두(a)), 수행하려고 하는 연산(첨가(+), 삭제(-), 교체(=)), 규정하려고 하는 허락(읽기(r), 쓰기(w), 실행(x))을 규정해 준다. 기호방식을 리용하는 선행실례에서는 그룹(g)에 대하여 쓰기(w)허락이 첨가되고 기타(o)에 대하여 실행(x)허락

이 제거(−)되였다.

아래에 chmod에서 흔히 리용되는 기호들을 개괄하였다.

영향을 받는 사람의 기호:

u	사용자가 영향을 받는다.
g	그룹이 영향을 받는다.
o	기타가 영향을 받는다.
a	모든 사용자들이 영향을 받는다.

수행할 조작:

+	허락을 첨가.
−	허락을 제거.
=	허락을 교체.

규정된 허락:

r	읽기허락.
w	쓰기허락.
x	실행허락.
u	사용자허락들을 복사.
g	그룹허락들을 복사.
o	기타의 허락들을 복사.

지령소개

여기서는 이 장에서 리용된 지령들을 묶어서 보여 주고 있다. 지령들은 UNIX변종들에서 차이나므로 일부 지령들에 대한 선택항목들과 다른 령역들에서 차이나는 점들이 있을수 있다. 그러나 아래의 지령들은 좋은 기준으로 리용될수 있다.

kill

kill − 프로쎄스에 신호를 보낸다.

---

kill(1)

이름

kill − 프로쎄스에 신호를 보낸다. 즉 프로쎄스를 종결시킨다.

형식

kill [-s signame] pid···

kill [-s signum] pid···

kill -l

퇴화된 방안들 :

kill -signame pid···

kill -signum pid···

해설

kill은 PID프로쎄스식별기호로 규정된 매 프로쎄스에 신호를 보낸다. 기정신호는 SIGTERM이다. kill은 표준적으로 신호 SIGTERM를 무시하지 않는 프로쎄스들을 종결시킨다. PID는 프로쎄스식별기호인 부호없는 또는 부의 옹근수인데 다음것들중의 하나일수 있다.

> 0	프로쎄스의 번호
= 0	프로쎄스그룹 ID가 송신자의 프로쎄스그룹 ID와 같은 특수한 체계프로쎄스들을 제외한 모든 프로쎄스들.
= -1	사용자가 알맞는 특권을 가진 특수한 체계프로쎄스들을 제외한 모든 프로쎄스들. 다른 경우에 실제적인 또는 효과적인 사용자ID가 송신하는 프로쎄스의 사용자 ID와 같은 특수한 프로쎄스들을 제외한 모든 프로쎄스들.
< -1	프로쎄스그룹 ID가 PID의 절대값과 같고 실제적인 또는 효과적인 사용자 ID가 송신하는 프로쎄스의 사용자와 같은 특수한 체계프로쎄스들을 제외한 모든 프로쎄스들.

프로쎄스번호들은 ps지령(ps(1)을 참고)과 어떤 쉘에서 리용할수 있는 내장(built-in)일감지령으로 찾을수 있다.

선택항목들

kill은 다음과 같은 선택항목들을 가진다.

-l(ell)　실현에 의해 지원되는 signame의 모든 값들을 렬거한다. 이 선택항목을 주면 신호들이 송신되지 않는다. 신호들의 기호적인 이름들은(SIG앞붙이가 없이)공백과 행바꾸기로 분리되

여 표준출구에 씌여 진다.

-s signame 규정된 신호이름을 송신한다. 기정은 SIGTERM, 번호는 15이다. signame은 대문자 또는 소문자로 규정될수 있으며 SIG 앞붙이가 붙을수 있다. 이 값들은 선택항목 -l을 리용하여 얻어 질수 있다. 기호적이름SIGNULL은 신호값0을 포함한다(아래의 신호이름의 번호들을 참고).

-s signum 규정된 10진신호번호를 송신한다. 기정은 15, SIGTERM이다(아래의 신호이름과 번호를 참고).

-signame(퇴화됨) -s signame과 동등하다.

-signum(퇴화됨) -s signum과 동등하다.

신호이름과 번호들

아래의 표에서는 말단으로부터 리용할수 있는 공통적인 신호들을 서술하고 있다. 전체 목록과 그 목록에 대한 완전한 서술을 보려면 머리부파일 <signal.h>와 안내항목 signal(5)를 보면 된다.

signum	signame	이 름	설 명
0	SIGNULL	Null	pid에로의 접근을 검사
1	SIGHUP	Hangup	종결될수 없다.
2	SININT	Interrupt	종결될수 없다.
3	SIGQUIT	Quit	주기억쏟기로 종결될수 없다.
9	SIGKILL	Kill	종결을 강요할수 있다.
15	SIGTERM	Terminate	종결을 강요할수 없다.
24	SIGSTOP	stop	프로쎄스를 정지시킬수 있다.
25	SIGTSTP	Terminal stop	프로쎄스를 정지시킬수 없다.
26	SIGCONT	Continue	정지된 프로쎄스를 실행시킨다.

SIGNULL(0)(null신호)는 오유검사를 진행하지만 신호를 실제적으로 보내지 않는다. 이것은 PID의 유효성 또는 존재성을 검열하는데 리용된다.

SIGTERM(15)(기정종결신호)는 수신프로쎄스에 의해 막힐수 있다. 수신자는 규칙적으로 shoutdown을 설정하거나 그 신호를 통채로 무시할수 있다.

SIGKILL(9)(제거신호)는 프로쎄스를 즉시 종결시킨다. SIGKILL은 막히거나 무시될수 없기때문에 SIGTERM에 응답하지 않는 프로쎄스를 종결시키는데 쓰일수 있다.

사용자가 알맞는 특권을 가지지 않는 한 수신프로쎄스는 송신프로쎄스의 사용자에게 속해야 한다.

특수한 경우로서 련속신호 SIGCONT는 송신프로쎄스와 동일한 작업기간에 실행되는 임의의 프로쎄스에 송신될수 있다.

귀환값

kill은 완결된후 다음 값들중의 하나를 돌려 준다.

0	매 PID연산인수에 맞는 프로쎄스가 적어도 하나 찾아 졌다. 그리고 규정된 신호는 그 프로쎄스에 대하여 성과적으로 처리되였다.
>0	오유가 발생하였다.

실례들

지령 :

```
kill 6135
```

는 프로쎄스번호6135를 종결시킬것을 신호한다. 이것은 프로쎄스에서 순조롭게 탈퇴할 기회를 준다(일시적파일들을 제거하고).

아래의 동등한 지령들

```
kill -s SIGKILL 6135
kill -s KILL 6135
kill -s 9 6135
kill -SIGKILL 6135
kill -KILL 6135
kill -9 6135
```

은 SIGKILL신호를 송신하여 프로쎄스번호6135를 급히 종결시킨다. 이것은 핵심부(kernel)가 그 프로쎄스를 즉시 종결시키도록 한다.

경고

프로쎄스가 전혀 자원을 분배 받을수 없도록 어떤 조작(I/O같은)에 머물러 있으면 실행이 허락될 때까지 그 프로쎄스를 제거할수 없다. 따라서 그 프로쎄스는 kill을 수행한후에도 없어 지지 않는다. 류사하게 종결된 프로쎄스들(ps(1)을 참고)은 실행을 이미 끝냈어야 하지만 선조프로쎄스들이 그것들을 얻을 때까지 체계상에 남아 있어야 한다(wait(2)를 참고).
kill을 리용하여 종결된 프로쎄스들에 보낸 신호들은 효력을 가지지 않는다. 일부 HP-UX가 아닌 다른 실현들은 kill을 쉘내장지령으로서만 제공한다.

종속성

이 안내항목은 외부지령 /usr/bin/kill과 POSIX쉘의 내장지령 kill을 서술한다(sh-posix(1)을 참고). C 와 Korn과 같은 다른 쉘들도 kill을 내장지령으로 제공한다(csh(1)과 ksh(1)을 참고). 그 내장지령들의 문장형식은 서로 다르다.

관련항목

csh(1), ksh(1), ps(1), sh(1), sh-bourne(1), sh-posix(1), kill(2), wait(2), signal(5)

표준일치

kill:SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

ksh

ksh - 지령프로그람작성언어

---

ksh(1)

이름

ksh, rksh - 쉘, 표준/제한된 지령프로그람작성언어

형식

ksh [+aefhikmnoprstuvx] [+o option] … [-c string] [arg …]
rksh [+aefhikmnoprstuvx] [+o option] … [-c string] [arg …]

해설

ksh는 말단 또는 파일로부터 읽은 지령들을 실행시키는 지령프로그람작성언어이다. rksh는 지령해석기 ksh의 제한된 방안인데 가입이름과 실행환경을 설치하는데 리용된다. 지령행선택항목들과 인수들에 대하여 상세한것을 알려면 뒤에서 서술되는 ksh호출과 특수한 지령부분 특히 지령 see를 보면 된다.

정의

메타기호

다음 기호들중의 하나이다.

; & ( ) | < > 행바꾸기 공백 tab

공백 tab 또는 빈자리

식별기호

문자 또는 밑줄로 시작되는 문자, 수자, 밑줄들의 렬이다. 식별기호들은 함수들에 대한 이름과 이름지어 진 파라메터로 리용된다.

단어 쉘언어의 문장론에 있는 기호들의 렬이다. 쉘은 매 지령을 읽고 요구되는 작용을 직접 또는 다른 봉사수단을 리용하여 수행한다.

특수지령

다른 프로쎄스를 창조함이 없이 쉘에 의해서 실행되는 지령이다. 이것을 **내장지령**(Built-In Command)라고도 부른다. 문서화된 측면효과를 제외하고 대부분의 특수지령들은 다른 봉사수단으로서 실행된다.

이 기호는 설명문의 시작으로 해석된다.

지령들

단순지령은 공백으로 분리된 단어들의 렬이다. 단어들의 앞에는 파라메터대입목록이 있을수 있다. 첫 단어는 실행되여야 할 지령의 이름을 규정한다. 아래에서 규정된것을 제외하고 나머지 단어들은 호출되는 지령에 인수로서 전달된다. 지령이름은 인수 0으로 전달된다(exec(2)를 참고).

단순지령의 값은 그 지령이 정상적으로 종결되였을 때 탈퇴상태이거나 비정상적으로 종결되였을 때 (8진)200+상태이다(상태값들에 대하여서는 signal(5)를 참고).

파이프선은 |으로 분리된 하나 또는 그이상의 지령들의 렬이다. 마지막지령을 제외한 매 지령의 표준출력은 파이프(pipe(2)를 참고)에 의하여 다음번 지령의 표준입력에 련결된다. 매 지령은 서로 다른 프로쎄스로서 실행된다. 쉘은 마지막지령이 종결되기를 기다린다. 파이프선의 탈퇴상태는 파이프선에 있는 마지막지령의 탈퇴상태이다.

목록은 하나 또는 그이상의 파이프선들이 ；, &, && 또는 ||에 의하여 분리되며 선택적으로 ；, & 또는 ｜&로 끝나는 렬이다. 이 다섯개 기호들중에서 ；, &, ｜&는 같은 우선권을 가진다. &&과 ||은 보다 높거나 같은 우선권을 가진다. 두반점(；)은 앞선 파이프선을 순차적으로 실행시키며 &는 앞선 파이프선을 비동기적으로 실행시킨다. (즉 쉘은 그 파이프선이 끝나기를 기다리지 않는다.) 기호 ｜&는 앞선 지령 또는 파이프선을 선조 쉘에로 설치된 두 통로파이프를 가지고 비동기적으로 실행시킨다. 파생된 지령의 표준입력과 출력은 후에 서술되는 특수한 지령 read와 print의 선택항목 -p를 리용하여 선조 쉘에 씌여 지거나 선조 쉘로부터 읽을수 있다. 기호 &&(｜｜)는 그 뒤에 있는 목록을 그앞의 파이프선이 0값(0아닌값)을 돌려줄 때에만 실행되게 한다. 목록에서 지령들의 경계를 정하는데 반두점대신 임의의 개수의 행바꾸기가 리용될수 있다.

지령은 단순지령이든가 다음것들중의 하나이다. 다르게 서술되지 않는 한 어느 한 지령이 돌려 주는 값은 그 지령에서 실행된 마지막단순지령의 값이다.

for identifier [in word …] do list done

for가 실행될 때마다 식별기호는 in word목록으로부터 취해 진 다음번 단어로 설정된다. in word …이 생략되면 for는 매 위치적파라메터모임에 대하여 do list를 한번 실행시킨다. 목록에 단어가 더는 없을 때 실행은 끝난다.

select identifier [in word …] do list done

select지령은 앞에 번호를 붙인 단어들의 모임을 표준오유(파일서술자 2)에 인쇄한다. in word …이 생략되면 위치적파라메터들이 대신 리용된다. PS3입력재촉문이 인쇄되고 표준입력으로부터 한개 행을 읽는다. 이 행이 렬거된 한개 단어의 번호로 이루어 지면 이 파라메터 identifier의 값은 그 번호에 대응하는 단어로 설정된다. 이 행이 비면 선택목록이 다시 인쇄되며 그렇지 않으면 파라메터 identifier의 값은 null로 설정된다. 표준입력으로부터 읽어 들인 행의 내용은 파라메터REPLY에 보관된다. list는 break 또는 end_of_file(eof)를 만날 때까지 실행된다.

case word in [[( )pattern [|pattern] …]list ;;]… esac

case지령은 word와 맞는 첫 pattern과 결합된 목록을 실행한다. pattern들의 형식은 파일이름생성을 위해 리용되는것과 같다.

if list then list [elif list then list] … [else list]fi

if뒤의 list가 실행된다. 만일 그것이 령탈퇴상태를 돌려 주면 첫 then뒤의 list가 실행되고 그렇지 않을 때에는 elif뒤의 list가 실행되며 만일 그 값이 령이면 then뒤의 list가 실행된다. 그것이 실패하면 else뒤의 list가 실행된다.

while list do list done

until list do list done

while지령은 while목록을 반복하여 실행시킨다. 그 목록안에 있는 마지막지령의 령탈퇴상태이면 do목록을 실행시키고 령이 아니면 순환은 종결된다. do목록에 있는 지령들이 실행되지 않으면 while은 령탈퇴상태를 돌려 준다. until은 순환종결검사를 부정하기 위하여 while대신에 리용될수 있다.

(list)

각이한 환경에서 list를 실행한다. 겹침을 위해 두개의 여는 괄호를 련달아 쓰는 경우에는 산수적인 평가를 피하기 위하여 공백을 삽입하여야 한다.

{ list ; }

각이한 환경이 아닌데서 list를 실행한다. {는 실마리어이며 뒤에 공백을 요구한다.

[[expression]]

식을 평가하고 진실이면 령탈퇴상태를 돌려 준다. 식에 대한 서술을 보려면 아래의 조건식을 보면 된다. [[와]]는 실마리어들이며 그것들과 식사이에는 공백이 있어야 한다.

function identifier {list;}

identifier ( ) {list;}

identifier로 참조되는 함수를 정의한다. 함수몸은 { }에 포함된 지령들의 목록이다.

time pipeline

pipeline이 실행되고 경과시간, 사용자시간, 체계시간이 표준오유에 인쇄된다.

아래의 실마리어들은(인용부호를 치지 않았을 때) 지령의 첫 단어로만 인식된다.

if then else elif fi case esac for while until do done { }
function select time [[ ]]

설명문

#는 행바꾸기전까기의 모든 기호들을 무시한다.

별명붙이기

매 지령의 첫 단어는 그 단어에 대한 별명이 정의되였다면 그 별명의 본문에 의하여 교체된다. 별명이름은 메타기호, 인용기호, 파일확장기호, 파라메터와 지령치환기호, =를 제외한 임의의 개수의 기호들로 구성된다. 교체기호렬은 우에서 렬거된 메타기호들을 포함하여 임의의 쉘스크립트를 포함할수 있다. 교체된 본문에 있는 매 지령의 첫 단어는 추가적인 별명을 위하여 검열된다. 별명값의 마지막기호가 공백이면 그 별명의 뒤에 있는 단어도 별명치환을 위하여 검사된다. 별명들은 특수한 내장지령들을 재정의하는데 리용되지만 우에서 렬거된 실마리어들을 재정의하는데는 리용되지 않는다. 별명들은 alias지령으로 창조되고 렬거되며 다른 쉘프로그람들에서 리용될수 있다. 또한 unalias지령으로 제거될수 있다. 다른 쉘프로그람들에서 리용될수 있는 별명들은 부분쉘들에서는 효력을 가지지만 쉘들을 따로따로 호출하기 위하여서는 재설치되여야 한다.

별명붙이기는 스크립트들이 실행될 때가 아니라 읽혀 질 때 수행된다. 그러므로 별명붙이기가 효력을 가지자면 별명들은 그 별명들을 창조하는 지령이 읽혀 지

기전에 수행되여야 한다.

별명들은 경로이름을 속기하는데 자주 리용된다. 별명붙이기수단에 주는 선택항목들은 별명의 값이 대응하는 지령의 경로이름으로 자동적으로 설정되도록 한다. 이 별명들을 추적된 별명이라고 부른다. 추적된 별명의 값은 그 식별기호가 처음으로 읽혀 질 때 정의되며 변수 PATH가 재설정될 때마다 그 정의가 무효로 된다. 별명들은 다음번 참조가 그 값을 재정의하도록 추적된다. 여러개의 추적된 지령들은 쉘안으로 콤파일된다. 지령 set의 선택항목 -h는 식별기호인 매 지령이름을 추적별명이름으로 변환한다.

아래의 다른 쉘프로그람들에서 리용될수 있는 별명들은 쉘안으로 콤파일되지만 비설정 또는 재정의될수 있다.

```
autoload='typeset  -fu'
false='let  0'
functions='typeset  -f'
hash='alias  -t  -'
history='fc  -l'
integer='typeset  -i'
nohup='nohup  '
r='fc  -e  -'
stop='kill  -STOP'
suspend='kill  -STOP  $$'
true=':'
type='whence  -v'
```

타일치환

별명치환이 수행된후 매 단어가 인용부호 없는 ~로 시작되는가를 검사한다. 만일 그렇다면 /까지의 단어가 /etc/passwd파일에 있는 사용자이름과 들어 맞는가를 검사한다. 정합이 진행되면 ~과 가입이름은 사용자의 가입등록부로 교체된다. 이것을 **타일치환**(Tilde Substitution)이라고 부른다. 정합이 진행되지 않으면 본래 본문은 교체되지 않는다. 홀로 또는 /앞에 있는 ~은 파라메터 HOME의 값으로 교체된다. + 또는 −가 뒤에 붙은 ~는 각각 파라메터PWD와 OLDPWD의 값으로 교체된다. 또한 타일치환은 파라메터대입의 값이 ~로 시작될 때 시도된다.

지령치환

괄호에 포함되여 있고 $가 앞에 붙은 지령($(<지령>)) 또는 웃반점에 포함되여 있는 지령(` <지령>`)으로부터의 출력은 단어의 부분 또는 전부로 리용될수 있다. 행바꾸기는 제거된다. 두번째 형식에서 인용부호사이에 있는 기호렬은 그 지령이 실행되기전에 특수한 인용기호들에 대하여 처리된다. 지령치환 $(cat파일)은 동등하지만 더 빠른 $(파일)로 교체될수 있다. I/O방향바꾸기를 수행하지

않는 특수한 지령들의 지령치환은 다른 프로쎄스를 창조하지 않고 수행된다. 그러나 함수의 지령치환은 그 함수와 함수안의 모든 지령들을 수행하기 위한 프로쎄스를 창조한다. 겹괄호안에 포함되여 있고 앞에 $가 붙은 산수식($((<식>)))은 그 겹괄호안에 있는 산수식의 값으로 교체된다.

파라메터치환

파라메터는 식별기호, 한개 또는 그이상의 수자들, 또는 *, @, #, ?, -, $, !들중의 하나이다. 이름지어 진 파라메터(식별기호로 표시되는 파라메터)는 한개의 값과 령 또는 그이상의 속성들을 가진다. 이름 지어 진 파라메터들에는 typeset지령을 리용하여 값과 속성들을 대입할수 있다. ksh에 의하여 지원되는 속성들은 후에 typeset지령과 함께 서술된다. 다른 프로그람들에서 리용될수 있는 파라메터들은 값과 속성들을 환경에 전달한다.

쉘은 제한된 1차원배렬기능을 가지고 있다. 배렬파라메터의 원소는 첨수에 의하여 참조된다. 첨수는 각괄호 [ ]안에 산수식이 놓여 있는 형태로 표시된다. 배렬에 값을 대입하려면 set -A name value를 리용한다. 모든 첨수들의 값은 0부터 1023사이에 있어야 한다. 배렬은 소개될 필요가 없다. 이름 지어 진 파라메터에 첨수를 붙여 참조할수 있으며 필요되면 배렬이 창조된다. 첨수가 없이 배렬을 참조하는것은 첫 원소를 참조하는것이다.

이름지어 진 파라메터의 값은 다음과 같은 형식으로 대입된다.

name=value[name=value]…

name에 대하여 -i로 옹근수속성이 설정되면 그 값은 아래에 서술된것과 같이 산수식으로 평가된다.

위치적파라메터(번호에 의해 표시되는 파라메터)들은 set지령으로 값을 받을수 있다. 파라메터 $0은 쉘이 호출될 때 인수 0으로 설정된다.

기호 $은 치환가능한 파라메터를 도입하는데 리용된다.

${parameter}

parameter의 값을 치환한다. parameter의 뒤에 이름의 부분으로 해석될수 없는 문자, 수자, 밑줄이 붙어 있거나 이름 지어 진 parameter에 첨수가 붙을 때 각괄호가 요구된다. parameter가 한개 혹은 그 이상의 수자들이라면 그것은 위치적파라메터이다. 한개이상의 수자로 된 위치적파라메터는 각괄호안에 넣어야 한다. parameter가 * 또는 @이면 $1로 시작되는 모든 위치적파라메터들이 치환된다. 배렬식별기호의 첨수가 * 또는 @이면 매 원소에 대한 값이 치환된다. 쉘은 $로부터 시작하여 모든 기호들을 동일한 단어의 부분으로 읽는다.

$｛#parameter｝

parameter가 * 또는 @이면 위치적파라메터들의 수가 치환되고 그렇지 않으면 parameter의 값의 길이가 치환된다.

$｛#identifier[*]｝

배렬식별기호에 있는 원소들의 개수를 치환한다.

$｛parameter：-word｝

parameter가 설정되여 있고 null이 아니면 그 값을 치환하고 그렇지 않으면 word를 치환한다.

$｛parameter：=word｝

parameter가 설정되여 있지 않고 null이면 그것을 word로 설정한다. 그 다음 그 parameter의 값을 치환한다. 위치적파라메터들은 이런 방식으로 대입될수 없다.

$｛parameter：?word｝

parameter가 설정되여 있고 null이 아니면 그 값을 치환하며 그렇지 않으면 word를 인쇄하고 쉘로부터 탈퇴한다. word가 생략되면 표준통보문이 인쇄된다.

$｛parameter：+word｝

parameter가 설정되여 있고 null이 아니면 word를 치환하며 그렇지 않으면 아무것도 치환하지 않는다.

$｛parameter # pattern｝

$｛parameter # # pattern｝

쉘pattern이 parameter값의 앞부분과 맞으면 이 치환의 값은 맞는 삭제된 몫이 있는 parameter의 값이며 그렇지 않으면 이 parameter의 값이 치환된다. 전자의 경우에는 가장 적게 맞는 패턴이 삭제되며 후자의 경우에는 가장 크게 맞는 패턴이 삭제된다.

$｛parameter%pattern｝

$｛parameter%%pattern｝

쉘pattern이 parameter값의 뒤부분과 맞으면 맞는 부분을 가진 파라메터의 값은 삭제되며 그렇지 않으면 parameter의 값을 치환한다. 전자의 경우에는 가장 작게 맞는 패턴이 삭제되며 후자의 경우에는 가장 크게

맞는 패턴이 삭제된다.

우에서 word는 치환된 기호렬을 리용되지 않는 한 평가되지 않는다. 따라서 아래의 실례에서 pwd는 d가 설정되지 않았거나 null일 때에만 실행된다.
echo ${d:-$(pwd)}

우의 식에서 두점(:)이 생략되면 쉘은 파라메터가 설정되지 않거나 null일 때에만 검사된다.

아래의 파라메터들은 쉘에 의하여 자동적으로 설정된다.

위치적파라메터들의 개수를 10진수로 준다.
- 쉘을 호출할 때 또는 set지령에 주는 선택항목이다.
? 마지막에 실행된 지령으로부터 귀한되는 10진수값.
$ 쉘의 프로쎄스번호
_ 초기에 _의 값은 환경에서 전달되였을 때 실행되는 쉘 또는 스크립트의 절대경로이름이다. 또한 _은 앞선 지령의 마지막인수로 대입된다. 이 파라메터는 비동기적인 지령에 대하여서는 설정되지 않는다. 이 파라메터는 또한 우편물을 검사할 때 맞는 MAIL파일의 이름을 유지하는데 리용된다.
! 마지막으로 실행되는 배경지령의 프로쎄스번호이다.

COLUMNS
이 변수가 설정되면 그 값은 쉘편집방식과 인쇄선택목록을 위한 편집창문의 너비를 정의하는데 리용된다. 창문환경에서 그 창문의 크기가 변경되면 쉘은 COLUMNS의 값을 변경시킨다.

ERRNO 가장 최근에 실패한 체계호출에 의하여 설정된 errno의 값이다. 이 값은 체계에 의존하며 오유수정에 리용된다.

LINENO 실행되고 있는 스크립트 또는 함수안에 있는 현재 행의 번호이다.

LINES 이 변수가 설정되여 있으면 그 값은 선택목록을 인쇄하기 위한 렬의 길이를 결정하는데 리용된다. 선택목록은 대략 LINES개의 행들중 2-3번째행이 채워 질 때까지 수직으로 인쇄한다. 창문환경에서 창문의 크기가 변경되면 쉘은 LINES의 값을 변경시킨다.

OLDPWD 지령 cd에 의하여 설정된 선행작업등록부이다.

OPTARG 특수한지령 getopts에 의하여 처리되는 마지막선택항목의 값이다.

OPTIND 특수한지령 getopts에 의하여 처리되는 마지막선택항목의 첨수이다.

PPID 쉘의 선조프로쎄스번호이다.

PWD 지령 cd 에 의하여 설정된 현 작업등록부이다.

RANDOM

이 파라메터가 평가될 때마다 0−33767사이에 분포된 우연수(옹근수)가 발생된다. 이 우연수렬은 수값을 RANDOM에 대입하여 초기화된다.

REPLY 이 파라메터는 선택명령과 파라메터가 없이 호출된 지령 read에 의하여 설정된다.

SECONDS

이 파라메터가 참조될 때마다 쉘이 호출된 때로부터의 시간을 초로 준다. 이 파라메터에 값이 대입되여 있으면 참조된 후에 귀환되는 값은 대입된 값에 대입된 때로부터의 초를 더한것이다.

아래의 파라메터들은 쉘에서 리용된다.

CDPATH 지령 cd의 탐색경로이다.

EDITOR 이 변수의 값이 emacs, gmacs, vi로 끝나고 변수VISUAL이 설정되여 있지 않으면 대응하는 선택항목이 on으로 된다.

ENV 이 파라메터가 설정되여 있으면 그 쉘이 호출될 때 수행되여야 할 스크립트의 경로이름을 발생시키기 위하여 그 변수에 대한 파라메터치환이 수행된다.

FCEDIT 지령 fc를 실행하기 위한 기정편집기의 이름이다.

FPATH 함수정의를 위한 탐색경로이다. 이 경로는 속성 -u가 있는 함수를 참조할 때와 지령이 발견되지 않을 때 탐색된다. 실행가능한 파일이 발견되면 그 파일은 현재 환경에서 읽혀 지고 실행된다.

IFS 내부적인 마당분리기호이다. 표준적으로는 지령 또는 파라메터치환으로부터 얻어 지는 지령단어들을 분리하는데 리용되는 공백, tab, 행바꾸기이다. 파라메터 IFS의 첫 기호는 "$*"치환을 위한 인수들을 분리시키는데 리용된다.

HISTFILE 쉘이 호출될 때 이 파라메터가 설정되여 있으면 그 값은 지령리력을 보관하기 위하여 리용되는 파일의 경로이름이다. 기정값은 $HOME/.sh_history이다. 만일 사용자가 알맞는 특권을 가지거나 HISTFILE 주어 져 있지 않으면 리력파일은 리용되지 않는다.

HISTSIZE 쉘을 호출하였을 때 이 파라메터가 설정되여 있으면 이 쉘에서 이미 입력된 지령들의 수는 이 수보다 크거나 같다. 표준값은 128이다.

HOME 지령 cd을 위한 기정인수(홈등록부)이다.

MAIL 이 파라메터가 우편파일의 이름으로 설정되여 있고 파라메터 MAILPATH가 설정되여 있지 않으면 쉘은 사용자에게 규정된 파일에 우편물이 도착하였다는것을 알려 준다.

MAILCHECK

이 변수는 파라메터 MAILPATH 또는 MAIL로 규정된 조작시간구

간에서 우편물이 도착하였는가를 쉘이 얼마나 자주 검사하여야 하는가를 규정한다. 기정값은 600s이다. 그 시간이 경과되면 쉘은 다음번 입력재촉문을 주기전에 검사한다.

MAILPATH
: 두점(:)으로 분리된 파일이름들의 목록이다. 이 파라메터가 설정되여 있으면 쉘은 마지막MAILCHECK초내에 규정된 파일을 변경시킬것을 사용자에게 알려 준다. 매 파일이름의 뒤에는 ?와 인쇄될 통보문이 놓인다. 이 경우에 그 통보문은 변경된 파일의 이름으로 정의된 파라메터 $_이 있는 파라메터 및 지령대입을 진행할것을 사용자에게 알린다. 기정통보문은 우편이 $_에 있다는것이다.

PATH
: 지령들에 대한 탐색경로이다. rksh가 실행되고 있으면 사용자는 PATH를 변경시킬수 없다.

PS1
: 이 파라메터의 값은 지령치환을 위해 $뒤에 공백기호를 붙인 1차입력재촉문기호렬의 정의에 의하여 확장된다. 1차입력재촉문기호렬에 있는 !는 지령번호로 교체된다. 입력재촉문에 !를 포함시키려면 !!를 리용하여야 한다.

PS2
: 2차입력재촉문기호렬이다. 기정으로는 >뒤에 공백이 붙는다.

PS3
: 선택순환안에서 리용되는 선택입력재촉문기호렬이다. 기정으로는 #?뒤에 공백이 놓인다.

PS4
: 이 변수의 값은 파라메터치환을 위하여 확장되며 실행추적의 매 행앞에 놓인다.

SHELL
: 쉘의 경로이름은 그 환경에서 유지된다. 그 쉘이 리용될 때 이 변수의 값이 토대이름의 r를 포함하면 그 쉘은 제한된다.

TMOUT
: 령보다 큰 값으로 설정되여 있는 경우에 지령이 PS1입력재촉문을 준 후 규정된 초시간내에 입력되지 않으면 쉘은 종결된다.

VISUAL
: 이 변수의 값이 emacs, gmacs, vi로 끝날 때 대응하는 선택항목을 리용한다.

쉘은 기정값들을 PATH, PS1, PS2, MAILCHECK, TMOUT, IFS에 준다. HOME, SHELL, ENV, MAIL은 전혀 쉘에 의하여 자동적으로 설정되지 않는다.

Blank해석

파라메터 및 지령치환후에 치환의 결과들은 마당분리기호들에 관하여 검사되며 그러한 기호들이 들어 있는 여러 인수들로 분할된다. ksh는 양적인 null인수들은 유지하지만 음적인 null기호들은 제거한다.

파일이름생성

매 지령단어는 선택항목 -f가 설정되지 않는 한 파일이름확장을 위한 패턴으로서 처리된다. 패턴들의 형식은 regexp(5)에 의해 정의된 패턴정합표기법이다. 단어는 패턴에 맞는 정돈된 파일이름들로 교체된다. 패턴에 맞는 파일이름이 없으면 단어는 변경되지 않는다.

regexp(5)에서 서술된 표기법외에 ksh는 하나 혹은 그이상의 패턴목록들이 서로 | 로 분리되여 만들어 진 합성패턴들을 인식할수 있다. 합성패턴들은 다음과 같은 형식으로 되여 있다.

?(패턴목록)	주어 진 패턴들중의 임의의것에 맞는다.
*(패턴목록)	주어 진 패턴들중의 령 또는 그이상에 맞는다.
+(패턴목록)	주어 진 패턴들중의 하나 또는 그이상에 맞는다.
@(패턴목록)	주어 진 패턴들중의 정확히 하나와 맞는다.
!(패턴목록)	주어 진 패턴들중의 하나를 제외한 임의의것에 맞는다.

인용문

우에서 렬거된 매개 메타기호는 특수한 의미를 가지며 인용되지 않는 한 단어의 종결을 의미한다. 기호의 앞에 \이 놓이면 그것은 인용문이다. \과 행바꾸기의 쌍은 무시된다. 두 웃반점사이(' ')에 놓여 있는 기호들은 인용문이다. 웃반점은 그안에 들어 있을수 없다. 웃두점사이(" ")에는 파라메터 및 지령치환이 나타나며 \은 \ , ` , " , $들을 위한 인용부호로 리용된다. $*와 $@은 인용되지 않을때 또는 파라메터대입값이나 파일이름으로 리용될 때 같은 의미를 가진다. 그러나 지령인수로 리용될 때 "$*"은 "$1d$3d…"과 동등하다. 여기서 d는 IFS파라메터의 첫 기호이다. 한편 "$@"은 "$1" "$3"…과 동등하다. 거꿀웃반점사이(` `)에서 \는 \ , ` , $들을 위한 인용부호로 리용된다.

실마리어 또는 별명, 함수이름 또는 지령이름을 인용부호안에 넣으면 그것들의 의미는 없어 진다.

산수적평가

옹근수산수식을 수행하기 위한 가능성은 특수한 지령 let로 제공된다. 계산은 긴 산수식을 리용하여 수행된다. 상수들은 [토대 #]n형식을 가진다. 여기서 토대는 산수적토대를 표현하는 2−36사이의 10진수이며 n은 그 토대의 어느 한 수이다. 토대가 생략되면 10이 리용된다.

산수식들은 C언어의 식과 동일한 문장론, 우선순위, 결합성을 리용한다. 연산자 ++, −−, ?:와 반점연산자들을 제외한 모든 형태의 연산자들을 리용할수 있다. 변수들은 파라메터치환형식을 리용하지 않고 산수식내에서 이름에 의해 참조된다. 변수가 참조될 때 그 변수의 값은 산수식으로 평가된다.

변수의 내부적인 옹근수표현은 typeset 전문지령의 선택항목 -i로 규정될수 있다. 산수식평가는 -i속성을 가진 변수에 대입된 값우에서 수행된다. 산수적토대를 규정하지 않은 경우 변수에로의 첫 대입이 산수적토대를 결정한다. 이 토대는 파

라메터치환이 진행될 때 리용된다.
많은 산수적연산자들이 인용부호를 요구하기때문에 다른 형식의 let지령이 마련되여 있다. "(("로 시작되는 임의의 지령에 대하여 "))"까지의 모든 기호들이 인용된 식으로 취급된다. 더 정확하게 ((…))는 let "…"와 동등하다.

입력재촉하기

쉘은 지령을 입력하기전에 PS1의 값으로 재촉한다. 임의의 시각에 행바꾸기가 입력되고 지령을 완성하기 위하여 입력이 더 필요된다면 2차재촉(PS2의 값)이 있게 된다.

조건식

조건식은 파일들의 속성들을 검열하고 기호렬들을 비교하기 위하여 [[로 구성된 지령과 함께 리용된다. 단어가르기와 파일이름생성은 [[와]]사이에 있는 단어들에 대하여서는 수행되지 않는다. 매 식은 아래의 1항 또는 2항식들중의 하나 또는 그이상으로 구성될수 있다.

-a file	file이 존재하면 진실
-b file	file이 존재하고 블로크전문파일이면 진실
-c file	file이 존재하고 기호전문파일이면 진실
-d file	file이 존재하고 등록부이면 진실
-f file	file이 존재하고 보통 파일이면 진실
-g file	file이 존재하고 그것의 setgid비트가 설정되여 있으면 진실
-h file	file이 존재하고 기호적인 련결이면 진실
-k file	file이 존재하고 sticky비트가 설정되여 있으면 진실
-n string	string의 길이가 령이 아니면 진실
-o option	option으로 이름지어 진 선택항목이 on이면 진실
-p file	file이 존재하고 fifo전문파일이거나 파이프이면 진실
-r file	file이 존재하고 현 프로쎄스에 의하여 읽기가능하면 진실
-s file	file이 존재하고 크기가 령보다 크면 진실
-t fildes	파일서술자번호마당들이 열려 져 있고 말단장치에 결합되여 있으면 진실
-u file	file이 존재하고 그것의 setuid비트가 설정되여 있으면 진실
-w file	file이 존재하고 현 프로쎄스에 의하여 쓰기가능하면 진실
-x file	file이 존재하고 현 프로쎄스에 의하여 실행가능하면 진실이다. 파일이 존재하고 등록부이면 현 프로쎄스는 그 등록부에서 탐색하기 위한 허락을 가진다.
-z string	string의 길이가 령이면 진실

-H file　file이 존재하고 은폐된 등록부이면 진실
-L file　file이 존재하고 기호적인 련결이면 진실
-O file　file이 존재하고 이 프로쎄스의 효과적인 사용자ID에 의하여 소유되여 있으면 진실
-G file　file이 존재하고 그것의 그룹이 이 프로쎄스의 효과적인 그룹 ID에 맞으면 진실
-S file　file이 존재하고 소케트이면 진실
file1 -nt file2　file1이 존재하고 file2보다 새것이면 진실
file1 -ot file2　file1이 존재하고 file2보다 낡은것이면 진실
file1 -ef file2　file1과 file2가 존재하고 동일한 파일을 가리키면 진실
string = pattern　string 이 pattern에 맞으면 진실
string !=pattern　string 이 pattern에 맞지 않으면 진실
string1 < string2　string1이 ASCII코드값에 기초하여 strimg2보다 앞에 놓이면 진실
string1 > string2　string1이 ASCII코드값에 기초하여 strimg2보다 뒤에 놓이면 진실
exp1 -eq exp2　exp1이 exp2와 같으면 진실
exp1 -ne exp2　exp1이 exp2와 같지 않으면 진실
exp1 -lt exp2　exp1이 exp2보다 작으면 진실
exp1 -gt exp2　exp1이 exp2와 크면 진실
exp1 -le exp2　exp1이 exp2보다 작거나 같으면 진실
exp1 -ge exp2　exp1이 exp2보다 크거나 같으면 진실

합성식은 이 기초적인 식들에 우선권이 감소하는 순서로 아래에 렬거된것들을 적용하여 구성된다.

(expression)　expression이 진실이면 진실.그룹식에 리용된다.
! expression　expression이 허위이면 진실
expression1 && expression2　expression1과 expression2가 둘 다 진실이면 진실
expression1 | | expression2　expression1 또는 expression2가 진실이면 진실

입력/출력

지령이 실행되기전에 입력과 출력은 쉘에 의하여 해석되는 특수한 표기법을 리용하여 방향전환될수 있다. 아래의것은 단순지령안의 어디에나 나타날수 있고 그앞과 뒤에 지령이 놓일수 있으며 호출되는 지령에 전달되지 않는다. 지령 및 파라메터치환은 아래에 언급된것을 제외한 단어 또는 수자가 리용되기전에 진행

될수 있다. 파일이름생성은 패턴이 하나의 파일에 들어 맞을 때에만 진행되며 blank해석은 수행되지 않는다.

<word 파일 word를 표준입력으로 리용한다(파일서술자 0).

>word 파일 word를 표준출력으로 리용한다(파일서술자 1).그 파일이 존재하지 않으면 창조된다. 그 파일이 존재하면 선택항목 noclobber은 on으로 되고 이때 오유가 발생된다. 그렇지 않을 때 그 파일의 길이는 령으로 된다.

>|word 선택항목 noclobber를 무시한다는것을 제외하고 >와 같다.

>>word 파일 word를 표준출력으로 리용한다. 그 파일이 존재하면 출력은 그 파일에 첨가되며 그렇지 않으면 그 파일이 창조된다.

<>word 파일 word를 읽기와 쓰기를 위하여 표준입력으로 연다.

<<[-]word 쉘입력은 word와 맞는 행까지 또는 end_of_file까지 읽는다. word우에서는 파라메터치환, 지령치환, 파일이름생성이 수행되지 않는다. 결과 문서는 표준입력으로 된다. word의 임의의 기호가 인용되였으면 그 문서의 기호들에 대한 해석은 진행되지 않는다. 그렇지 않으면 파라메터 및 지령치환이 진행되며 \행바꾸기는 무시된다. \는 기호 \ , $, ‘와 word의 첫 기호를 인용하는데 리용된다. - 가 <<에 첨가되면 모든 tab들이 단어 또는 문서로부터 제거된다.

<&digit 표준입력은 파일서술자 digit로부터 복사된다.

>&digit 표준출력은 파일서술자 digit에 복사된다.

<&- 표준입력은 닫겨 져 있다.

>&- 표준출력은 닫겨 져 있다.

<&p 상대프로쎄스로부터의 입력은 표준입력에 이동된다.

>&p 상대프로쎄스로부터의 출력은 표준출력에 이동된다.

우에 있는것들중의 하나에 수자가 앞에 붙으면 그것은 인용된 파일서술자번호이다. 실례로

…2>&1

은 파일서술자 2가 파일서술자 1의 중복으로서 쓰기를 위해 열려야 한다는것을 의미한다. 쉘은 규정된 파일서술자와 결합된 현재 열린 파일에 관하여 파일서술자들을 참조하면서 방향전환을 평가하므로 방향전환순서는 중요한 문제이다. 실례로

…1>frame 2>&1

은 우선 파일서술자 1(표준출력)을 파일이름에 대입하고 그 다음 파일서술자2(표준오유)를 파일서술자 1에 대입한 파일 즉 frame에 대입한다. 다른 한편 방향전환의 순서가 다음과 같이 바뀌면

…2>&1 1>frame

파일서술자 2는 현재 표준출력(다른 대입이 전달되지 않는 한 사용자말단)에 대입된다. 파일서술자 1은 파일서술자 2의 대입을 변경시키지 않고 파일 frame에 다시 대입된다.

호상작용하는 프로쎄스들의 입력과 출력은 다른 지령들을 허락하는 번호붙은 파일서술자에로 옮겨 질수 있으며 우의 방향전환연산자들을 리용하여 써넣거나 읽을수 있다. 현 프로쎄스의 입력이 번호 붙은 파일서술자에로 옮겨 지면 다른 프로쎄스가 시동될수 있다. 지령뒤에 &가 붙고 일감조종이 동작하지 않고 있으면 그 지령에 대한 기정표준입력은 빈파일/dev/null이고 그렇지 않을 때에는 지령의 실행을 위한 환경이 입출력설명서에 의해 조작된대로 호출되는 쉘의 파일서술자들을 포함한다.

환경

환경(envirn(5)를 참고)은 일반적인 인수목록처럼 실행되는 프로그람에 전달되는 이름−값의 쌍들로 된 목록이다. 이름들은 식별기호들이여야 하며 값들은 기호렬이여야 한다. 쉘은 환경과 여러가지 방식으로 호상 작용한다. 호출될 때 쉘은 환경을 검사하고 찾아 진 매 이름에 대한 파라메터를 창조하며 그것에 대응하는 값을 주고 다른 프로그람에서 리용될수 있다는것을 표식한다. 실행된 지령들은 환경을 그대로 넘겨 받는다. 만일 사용자가 이 파라메터들의 값을 변경시키거나 지령 export 또는 typeset -x 를 리용하여 새것을 창조하면 그 값들은 환경의 일부분으로 된다. 실행되는 지령이 리용하는 환경은 쉘로부터 넘겨 받은 이름-값의 쌍들로 구성된다.

단순지령 또는 함수를 위한 환경은 하나 혹은 그이상의 파라메터대입을 붙여 확장시킬수 있다. 파라메터대입인수는 식별기호=값의 형태를 가진다. 실례로

TERM=450 cmd args

과

(export TERM; TERM=450; cmd args)

들은 동등하다.

선택항목 -k가 설정되여 있으면 모든 파라메터대입인수들이 그 환경에 배치된다(그것들이 지령이름뒤에 나타나지 않는 한).

아래의 echo명령문은 a=b c를 인쇄한다. 선택항목 -k가 설정된 후에 두번째 echo 명령문은 c만을 인쇄한다.

```
echo a=b c
set -k
echo a=b c
```

이 특성은 쉘의 선행방안을 위하여 씌여 진 스크립트에서 리용될수 있으며 새 스크립트에서는 리용되지 않는다.

함수들

실마리어 function는 쉘함수들을 정의하는데 리용된다. 쉘함수들은 내부적으로 읽혀 지고 보관된다. 별명이름들은 함수가 읽혀 질 때 지워 진다. 함수들은 위치적변수로 전달되는 인수들을 가지고 지령처럼 실행된다.

함수들은 함수의 지령치환이 새로운 프로쎄스를 창조한다는것을 제외하고 호출자와 동일한 프로쎄스에서 실행된다. 함수들은 모든 파일들과 현존 작업등록부를 호출자와 공유한다. 호출자에 의하여 생겨나는 **함정**(Trap)들은 함수내에서 기정작용으로 재설정된다. 만일 어느 한 함수가 함정조건을 포착하지 못하거나 무시한다면 그 함수는 종결되며 그 조건은 호출자에게 전달된다. 함수안에 있는 EXIT모임상의 함정은 그 함수가 호출자의 환경에서 완결된 후에 실행된다. 변수들은 보통 호출하는 프로그람과 호출되는 함수사이에서 공유된다. 그러나 함수내에서 리용되는 typeset전문지령은 현 함수와 그것에 의하여 호출되는 모든 함수들에서 효력을 가지는 국부적변수들을 정의한다.

전문지령 return은 함수호출로부터 귀환하기 위하여 리용된다. 함수안에서의 생겨 나는 오유들은 호출자에게 조종을 넘긴다.

함수식별기호들은 typeset지령의 선택항목 +f로 렬거할수 있다. 함수식별기호들과 함수들의 결합된 본문은 선택항목 -f로 렬거할수 있다. unset지령의 선택항목 -f로 함수정의를 무효로 할수 있다.

쉘이 쉘스크립트를 실행할 때 함수의 설정은 무효로 된다. typeset지령의 선택항목 -xf는 함수를 쉘을 재호출하지 않고 실행되는 스크립트에서 리용할수 있게 한다.

일감들

set지령의 선택항목 monitor가 on이면 대화형 쉘은 일감을 파이프선과 결합시킨다. set지령은 지령 jobs에 의하여 인쇄되는 현재 일감들의 표를 유지하며 일감들에 작은 옹근수들을 할당한다. 일감이 &에 의하여 비동기적으로 시동되면 쉘

은 다음과 같은 행을 인쇄한다.

[1] 1234

이것은 일감번호 1이 비동기적으로 시동되였으며 프로쎄스번호가 1234인 한개 프로쎄스를 가진다는것을 가리킨다.

만일 사용자가 어느 한 일감을 실행시키면서 그 무엇인가를 하려고 한다면 현 일감에 STOP신호를 보내기 위하여 정지기호(ctrl-z)를 입력한다. 그러면 쉘은 그 일감이 정지되였다는것을 가리키고 다른 입력재촉문를 인쇄한다. 이 일감의 상태는 지령 bg에 의하여 변경되여 배경으로 넘어 가고 다른 지령들을 실행시킬수 있다. 또한 지령 fg를 리용하여 일감을 전경에로 되돌릴수 있고 재시동시킬수 있다. ctrl-z는 즉시 효력을 나타내며 중단과 류사하다.

배경에서 실행되는 일감은 말단으로부터 읽으려고 할 때 정지된다. 배경일감은 표준적으로 출력을 생성할수 있게 허락되여 있으나 지령 stty tostop을 주어 출력을 할수 없게 할수 있다. 만일 사용자가 선택항목 tty을 설정하면 배경일감은 출력을 생성하려고 할 때 정지된다.

쉘에서 일감들을 지적하는 방법은 여러가지이다. 일감은 그 일감에 있는 임의의 프로쎄스의 PID로 또는 다음것들중의 하나로 지적될수 있다.

%number	주어 진 번호를 가진 일감
%string	지령행이 string으로 시작되는 임의의 일감
%?string	지령행이 string을 포함하는 임의의 일감
%%	현재일감
%+	%%와 동등하다.
%-	선행일감

쉘은 프로쎄스가 상태를 변경하였을 때 즉시 알아 차린다. 쉘은 일감이 정지되고 더 전진할수 없게 되였을 때 사용자에게 알려 준다.
일감들이 실행되고 있을 때 쉘에서 탈퇴하거나 정지하려고 한다면 실행되고 있는 일감을 정지시켰다는 경고가 발생된다. 일감들을 식별하려면 지령 jobs을 리용한다. 다시 탈퇴하려는 시도는 정지된 일감을 종결시킨다. 즉 쉘은 경고를 두번째로는 발생시키지 않는다.

신호들

호출된 지령에 대하여 그 지령에 &를 붙이고 선택항목 monitor가 off이면 INT와 QUIT신호들은 무시되며 그렇지 않을 때 신호들은 쉘에 의하여 선조로부터 넘겨 받는 값을 가진다(신호 11은 제외한다.).

실행

지령이 실행될 때마다 치환이 진행된다. 지령이름이 아래에 렬거된 특수한 지령들중의 하나와 맞으면 그 지령은 현 쉘프로쎄스내에서 실행된다. 그다음 ksh는 그 지령이 사용자정의함수와 맞는가를 결정하기 위하여 지령이름을 검사한다. 만일 맞는다면 ksh는 위치적파라메터를 보관하고 그것들을 함수호출의 인수로 설정한다. 위치적파라메터 0은 함수이름으로 설정된다. 함수가 완결되거나 귀환되면 ksh는 위치적파라메터목록을 보관하고 함수안의 EXIT에 대한 trap모임을 실행한다. 함수의 값은 마지막에 실행된 지령의 값이다. 함수는 현 쉘프로쎄스에서 실행된다. 지령이름이 전문지령 또는 사용자정의함수가 아니면 ksh는 한 프로쎄스를 생성하고 exec를 리용하여 그 지령을 실행시키려고 한다.

쉘파라메터 PATH는 그 지령을 포함하는 등록부에 대한 탐색경로를 정의한다. 대신 탐색해야 할 등록부이름은 두점(：)으로 분리한다. 기정경로는 /usr/bin이다. 현 등록부는 기호 =뒤에서 두점들사이에 또는 경로목록의 끝에 나타날수 있는 빈 경로이름에 의하여 규정된다. 지령이름이 /를 포함하면 탐색경로는 리용되지 않으며 그렇지 않을 때에는 경로에 있는 실행가능한 파일을 찾기 위하여 모든 등록부를 탐색한다. 그 파일이 실행허락을 가지지만 등록부가 아니거나 실행가능한 대상코드파일이 아니면 그 파일은 해석기를 위한 자료파일인 스크립트파일일수 있다. 그 스크립트파일의 첫 두개 기호가 #!이면 exec는 그 뒤에 해석기경로이름을 요구한다. 그 다음 exec는 규정된 해석기를 전체 스크립트파일을 읽기 위한 다른 프로쎄스로 실행시킨다. exec에 대한 호출이 실패하면 /usr/bin/ksh가 그 스크립트파일을 해석하기 위하여 파생된다. 모든 반출되지 않은 별명들, 함수들, 이름 지어 진 파라메터들은 이 경우에 제거된다. 쉘지령파일이 읽기허락을 가지지 않거나 setuid와 setgid비트들이 그 파일에 관하여 설정되여 있으면 쉘은 허락을 설정하고 열린 파일로서 전달된 쉘지령파일을 가지고 쉘을 실행시키기 위하여 대리자를 실행시킨다. 괄호안에 있는 지령은 다른 곳에서 리용될수 없는 량들을 제거하지 않고 부분쉘에서 실행된다.

지령재입력

말단장치로부터 입력된 마지막HISTSIZE개(기정128) 지령들의 본문은 리력파일에 보관된다. 파일 $HOME/.sh_history는 변수 HISTFILE가 설정되지 않았거나 쓰기가능한 경우에 리용된다. 쉘은 동일하게 이름 지어 진 HISTFILE을 리용한 모든 호상작용하는 쉘들의 지령들을 리용할수 있다. 이 파일의 일부분을 렬거하거나 편집하기 위하여 전문지령 fc가 리용된다. 편집되거나 렬거되여야 할 그 파일의 일부분은 번호 또는 지령의 첫 기호, 기호들을 주는것으로써 선택될수 있다. 한개의 지령 또는 지령들의 범위가 규정될수 있다. fc의 인수로 규정된 편집프로그람이 없으면 FCEDIT파라메터의 값이 리용된다. FCEDIT가 정의되지 않았으면 /usr/bin/ed가 리용된다. 편집기를 탈퇴하면 편집된 지령이 인쇄되고 다시 실행된다. 편집기이름 −은 편집단계를 넘기고 그 지령을 재실행시키기 위하

여 리용된다. 이 경우에 old=new형식의 치환파라메터는 실행하기전에 그 지령을 변경시키기 위하여 리용될수 있다. 실례로 r가 fc -e -로 별명 지어 져 있는 경우에 r bad=good을 입력하면 문자 c로 시작되는 가장 최근의 지령이 재실행되고 제일 처음에 출현하는 기호렬 bad가 기호렬 good으로 교체된다.

umask

umask - 창조마스크를 설정하거나 현시한다.

---

umask(1)

이름

umask - 파일방식창조마스크를 설정하거나 현시한다.

형식

마스크설정　　umask mask
마스크현시　　umask [-S]

해설

umask지령은 파일방식창조마스크의 값을 설정하거나 현재 방식을 현시한다. 마스크는 후에 창조되는 파일들에 대한 파일방식(허락)비트들의 초기값에 영향을 준다.

파일방식창조마스크의 설정

umask지령은 현재 쉘의 실행환경에 대한 새로운 파일방식창조마스크를 설정한다. 마스크는 기호값 또는 수값일수 있다. 기호마스크는 마스크허락비트들을 개별적으로 또는 그룹으로 변경시키는 유연한 방식을 준다. 수값마스크는 모든 허락비트들을 한번에 규정한다. 마스크가 규정되여 있지 않을 때 표준출력에 기입되는 출력은 없다.

기호마스크값

기호마스크는 현 파일방식창조마스크를 교체하거나 변경시킨다. 이 마스크는 다음과 같은 형식으로 규정된다. 공백은 허용되지 않는다.

[who] [operator] [permissions][, …]

마당들은 다음의 값들을 가질수 있다.

who　　다음 문자들중의 한개 또는 여러개일수 있다.

u　　사용자(소유자)에 대한 허락들을 변경시킨다.

u　사용자(소유자)에 대한 허락들을 변경시킨다.
g　그룹에 대한 허락들을 변경시킨다.
o　기타에 대한 허락들을 변경시킨다.
a　모두에 대한 허락들을 변경시킨다(a=ugo).

operator　다음 기호들중의 한개일수 있다.
＋　who에 대하여 이미 존재하는 마스크에 허락들을 첨가한다.
-　who에 대하여 이미 존재하는 마스크에서 허락들을 삭제한다.
=　who에 대하여 이미 존재하는 마스크를 허락들로 교체한다.

permissions　다음 문자들중의 한개 또는 여러개일수 있다.
r　읽기허락
w　쓰기허락
x　실행/탐색허락

한개 또는 두개의 마당들이 생략되면 다음의 표의 내용을 적용한다.

입력된 형식	효 과	입력	동등
who	who에 대한 현재허락을 삭제한다.	g	g=
operator	작용없다.	-	(none)
permissions	a+permissions와 같다.	rw	a+(rw)
who=who에 대한 현재허락을 삭제	u=	u=	
who+	작용 없다.	u+	(none)
who-	작용 없다.	u-	(none)
whopermissions	who=permisions와 같다.	ux	u=x
operatororpermissions	operatororpermissions와 같다.	-rw	a-rw

수값마스크

수값마스크는 현재의 파일방식창조마스크를 교체한다. 이것은 다음의 방식비트들의 론리적인 합으로부터 구성되는 부호 없는 8진수로 규정된다.

0400 (a=rwx, u-r)　소유자에 의한 읽기
0200 (a=rwx, u-w)　소유자에 의한 쓰기
0100 (a=rwx, u-x)　소유자에 의한 실행(등록부에서 탐색)
0040 (a=rwx, g-r)　그룹에 의한 읽기
0020 (a=rwx, g-w)　그룹에 의한 쓰기
0010 (a=rwx, g-x)　그룹에 의한 실행/탐색

0004 (a=rwx, o-r)　기타에 의한 읽기
0002 (a=rwx, o-w)　기타에 의한 쓰기
0001 (a=rwx, o-x)　기타에 의한 실행/탐색

현재의 마스크값의 현시

현재의 파일방식창조마스크값을 현시하려면 다음 지령들중의 하나를 리용한다.
umask -S 현 파일방식창조마스크를 다음과 같은 기호적인 형식으로 인쇄한다.

u=[r] [w] [x] , g=[r] [w] [x] , o=[r] [w] [x]

기호 r(읽기), w(쓰기), x(실행/탐색)들은 u(사용자/소유자), g(그룹), o(기타)에 대하여 마스크에서 지워 진 비트들을 표현한다. 다른 비트들은 설정되여 있다.

uamsk　현재의 파일방식창조마스크를 8진수로 인쇄한다.
수값이 0으로 된 비트들은 기호값의 현시된 r, w, x허락기호들에 대응한다. 수값이 1로 된 비트들은 기호값에서 빠진 허락기호들에 대응한다.
실현에 따라 1−4개 8진수들이 현시된다. 첫 수자는 항상 0이다. 오른쪽에 있는 세개 수자들은 마스크에 설정되여 있거나 지워 진 비트들을 표현한다.

이 두 형식들은 후에 umask지령에서 마스크를 설정하기 위하여 마스크인수로 리용될수 있는 출력을 생성한다.

일반조작

새로운 파일이 창조될 때(creat(2)를 참고) 파일방식창조마스크에서 설정된 매 비트는 파일방식에 있는 대응하는 허락비트를 지운다. 반대로 마스크에서 지워지는 비트들은 대응하는 파일방식들이 새롭게 창조되는 파일들에서 불가능하게 한다.

실례로 마스크 u=rwx, g=rx, o=rx(022)는 그룹과 기타에 대한 쓰기허락들을 불가능하게 한다. 결국 ls -l지령에 의하여 보여 준 -rwxrwxrwx(777)와 같은 파일방식으로 창조된 파일들은 방식 -rwxr-xr-x(755)로 된다. 한편 파일방식 -rw-rw-rw(666)으로 창조된 파일들은 방식 -rw-r--r--(644)로 된다.

파일방식창조마스크는 set-user-id, set-group-id, sticky비트들에는 영향을 주지 않는다.

파일방식창조마스크는 또한 chmod지령에 의하여 리용된다(chmod(1)을 참고).

umask는 현재 쉘창조환경에 영향을 주기때문에 그것은 일반적으로 내장지령으로 되여 있다.

umask는 다음과 같은 부분쉘 또는 다른 봉사수단실행환경에서 호출된다.

(umask 002)
nohup umask…
find .-exec umask…

이것은 호출하는 환경의 파일방식창조마스크에 영향을 주지 않는다.

기정마스크는 u=rwx, g=rwx, o=rwx(000)이다.

귀환값

umask는 다음 값들중의 하나를 가지고 귀환한다.

0	파일방식창조마스크가 성과적으로 변경되였거나 주어 진 마스크인수가 없다.
>0	오유가 발생하였다.

실례들

이 실례들에서 매개 행은 동일한 과제를 수행하기 위한 다른 방법을 보여 주고 있다.

파일소유자에 대하여서는 읽기와 쓰기허락을, 기타에 대하여서는 읽기허락을 주기 위한 umask값을 설정하자(ls -l은 새로 창조된 파일에 대하여 -rw-r--r--를 현시한다.).

umask u=rwx, g=rx, o=rx	기호방식
umask a=rx, u+w	기호방식
umask 022	수값방식

파일소유자에 대하여서는 읽기 및 쓰기허락을, 동일한 그룹에 있는 다른 사용자들에 대하여서는 read-only를 생성하기 위한 umask값을 설정하자.

umask a-rwx, u+rw, g+r	기호방식
umask137	수값방식

모든 사람들에게 읽기, 쓰기, 실행허락을 거부하기 위한 umask값을 설정하자.

umask a=   기호방식
umask 777   수값장식

매 사람에 대한 현 마스크에 쓰기허락을 첨가하자(동등한 수값방식은 없다.).

umask   a+w   기호방식

경고

사용자(소유자)를 위한 읽기 또는 쓰기접근을 방지하는 마스크를 설정하면 일시적파일들을 창조하는 편집기와 같은 프로그람들이 실행되지 않는다. 왜냐하면 그 프로그람들은 파일자료에 접근할수 없기때문이다.

종속성

umask지령은 독립적으로 실행가능한 파일(/usr/bin/umask)과 내장쉘지령으로 실현된다.

POSIX쉘과 독립파일

모든 특징들이 지원된다(sh-posix(1)을 참고).

Korn쉘

선택항목 -S는 Korn쉘내장지령에서 지원되지 않는다. 수값마스크현시는 가장 작은 두개 수자들을 리용한다.

C쉘

선택항목 -S와 기호마스크값들은 C쉘내장지령에서 지원되지 않는다. 수값마스크현시는 가장 작은 한개 수자를 리용한다.

Bourne쉘

선택항목 -S와 기호마스크값들은 Bourne쉘내장지령에서 지원되지 않는다. 수값마스크현시는 항상 4개 수자들로 이루어 진다.

관련항목

chmod(1), csh(1), ksh(1), sh-posix(1), sh(1), chmod(2), creat(2), umask(2)

표준일치

umask: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제１１장. C 쉘에 대한 개괄

각이한 쉘

대부분의 UNIX 변종들에는 여러개의 쉘들이 있다. 쉘을 통하여 체계를 쉽게 리해할 수 있기때문에 사용자들에게 있어서 쉘은 매우 중요하다. 쉘들에서 지령들을 주고 사용자환경들을 조종하며 지령파일들과 쉘프로그람들을 기입하는 등 UNIX 변종들에서 쉘들을 리용하는 방법은 류사하다. 쉘에서는 대체로 많은 시간이 소비되기때문에 여러개의 각이한 쉘들을 리해하고 그것들중의 하나를 선택해야 한다. Bash 쉘과 같은 특수한 쉘들은 UNIX 변종들에서 그리 차이나지 않는다. 쉘들은 자체의 고유한 특징들을 가진다. 일반적으로 사용자들은 어느 한개 쉘을 특별히 좋아 하게 된다. 모든 쉘들은 기능적으로 류사하며 그것을 안 다음에는 리용하기가 재미 있다. 대부분의 UNIX 변종들에서 체계관리자는 사용자들을 등록할 때 여러개의 각이한 쉘들중에서 하나를 선택할수 있으므로 사용자들이 실행시키는 쉘들을 유연하게 처리할수 있다. 일반적으로 체계관리자들은 체계관리를 쉽게 하기 위하여 사용자들이 동일한 쉘을 리용할것을 요구한다. 그러나 체계관리자들은 사용자가 체계에 있는 특정한 쉘을 리용해야 할 리유가 있으면 그것에 대한 사용자의 요구를 들어 주기도 한다. 이 장에서는 C 쉘을 취급한다. Bash 쉘과 Korn 쉘은 앞장들에서 취급되였다.

C 쉘에 대한 개괄

C 쉘은 UNIX 와의 사용자대면부를 제공하는데 있어서 다른 쉘들과 류사하다. C 쉘은 다음과 같은 3 가지 방식으로 리용될수 있다.

- 대화적으로 지령행에 지령들을 입력한다.
- 흔히 리용되는 지령모임들을 지령파일에 그룹화하고 그 파일의 이름을 입력하여 실행시킨다.
- C 쉘의 구조화된 프로그람작성기술을 리용하여 C 쉘프로그람들을 작성한다.

이 3 가지 기술들은 자주 리용되는 순서로 렬거되였다. 우선 체계에 가입하고 대화적지령들을 리용한다. 그다음 흔히 리용되는 지령들을 그룹화하고 그것을 하나의 지령으로 실행시킨다. 끝으로 쉘스크립트들을 창조한다.

이 장에서는 가입과 대화적지령들 그리고 C 쉘을 리용하는 유용한 방법들을 취급한다.

이 장에 있는 대부분의 실례들은 Solaris 와 HP-UX 체계의 실례들이다. 이 장에서 취급되는 체계와 류사한 다른 체계들에서도 C 쉘을 설치하고 리용할수 있다. 쉘의 설치가 체계관리자에 의하여 수행되므로 이 장에서 취급되지 않는 다른 C 쉘에서는 차이가 있을수 있다. 그러나 일반적으로 C 쉘의 조작은 모든 체계들에서 류사하다.

지령주기

체계에 가입한 다음 사용자가 수행하는 첫 작업은 입력재촉문에 따라 지령을 주는것이다. 처음으로 주어야 할 지령은 ls -al 이다. 이 지령이 실행되면 다음과 같은 파일목록이 생겨 난다.

```
sys1 7: ls -al
total 10
drwxr-x---   2  martyp2  users    96  May  5  09:34  .
drwxr-xr-x  10     root   root  1024  May  5  10:38  ..
-rw-r--r--   1  martyp2  users   814  May  5  09:34  .cshrc
-rw-r--r--   1  martyp2  users   347  May  5  09:34  .exrc
-rw-r--r--   1  martyp2  users   341  May  5  09:34  .login
-rw-r--r--   1  martyp2  users   446  May  5  09:34  .profile
sys1 8:
```

C 쉘의 입력재촉문은 체계이름(sys1)과 그뒤에 지령번호 그리고 두점(:)으로 이루어진다. ls -al 은 사용자령역에서 C 쉘에 련관되는 두개 파일들 즉 .cshrc 와 .login 을 보여준다.

그림 11-1 에서는 .cshrc 의 내용을 보여 주고 있다.

```
# Default user .cshrc file (/usr/bin/csh initialization).

# Usage:  Copy this file to a user's home directory and edit it to
# customize it to taste. It is run by csh each time it starts up.

# Set up default command search path:
#
# (For security, this default is a minimal set.)

  set path=( $path )

# Set up C shell environment:

  if ( $?prompt ) then      # shell is interactive.
    set history=20  # previous commands to remember.
    set savehist=20 # number to save across sessions.
    set system='hostname'   # name of this system.
    set prompt = "$system \!:"     # command prompt.
```

```
  # Sample alias:

  alias   status '(date; bdf)'

  # More sample aliases:

  alias   d       dirs
  alias   pd      pushd
  alias   pd2     pushd +2
  alias   po      popd
  alias   m       more
endif
```

그림 11−1. .cshrc 의 실례

그림 11−2 에서는 .login 의 내용을 보여 주고 있다.

```
# @(#) $Revision: 72.3 $

# Default user .login file ( /usr/bin/csh initialization)

# Set up the default search paths:
set path=( $path )

#set up the tenrinal
eval `tset -s -Q -m ':?hp'`
stty erase "^H" kill "^U" intr "^C" eof "^D" susp "^Z" hupcl ixon ixoff
tostop
tabs

# Set up shell environment:
set noclobber
set history=20
```

그림 11−2. .login 의 실례

.cshrc 파일

체계에 가입한후에 일어 나는 현상들은 UNIX 체계들마다 다르다. 많은 체계들에서는 파일 .cshrc 이 먼저 C 쉘에 의하여 읽혀 지고 실행된다. 사용자는 리용하려고 하는 지령행입력재촉문을 규정하며 리력목록을 초기화하고 별명들을 정의하기 위하여 .cshrc 파일을 변경할수 있다. 아래에서는 그림 11−1 에서 보여 준 .cshrc 파일이 지령행입력재촉문을 규정하며 리력목록을 초기화하고 별명들을 정의하는 방법을 보여 준다. 먼저 아래의 항목에서 .login 파일을 간단히 고찰해 보자.

.login 파일

많은 UNIX 체계들에서 파일 .login 은 파일 .cshrc 다음에 읽혀 진다. 우의 실례에는 설치와 관련하여 실행된 두개의 지령만이 있다. 첫번째는 TERM 환경변수를 설정하는 지령 tset 이다. tset 앞에 놓인 eval 은 C 쉘이 **후손**(child) 프로쎄스를 창조함이 없이 tset 와 그것의 인수들을 실행시킨다는것을 의미한다. 이것은 tset 가 쓸모 없는 부분쉘대신에 현 쉘에 있는 환경변수들을 설정할수 있게 한다. stty 지령은 말단 I/O 선택항목들을 설정하는데 리용된다. 이 두 설정지령들은 쉘변수들을 정의하는데 리용된다. noclobber 는 이미 존재하는 파일우에 기입하게 하는 방향바꾸기를 허용하지 않는다. /tmp/processes 와 같은 이미 존재하는 파일우에 기입하려고 한다면 그 파일이 존재한다는 통보문을 받는다.

```
sys1 1: ````ps -ef > /tmp/processes
/tmp/processes: File exists
```

">"은 ps 의 출력을 화면에로가 아니라 /tmp/processes 에로 보낸다는것을 의미한다. 그러나 파일 /tmp/processes 는 ps -ef 의 출력으로 덧쓰기될수 없다. 왜냐하면 /tmp/processes 가 이미 존재하며 noclobber 라고 부르는 환경변수가 설정되여 있기때문이다. noclobber 가 설정되여 있으면 이 파일에로의 출력은 진행될수 없다. 이것은 이미 존재하는 파일이 우연히 지워 지는것을 방지하는데 리용되는 기능이다. 방향바꾸기의 형식은 여러가지일 때만 이 장의 마감에서 취급하기로 한다.

.cshrc 안의 리력목록을 초기화

C 쉘은 실행된 지령들의 리력목록을 가질수 있다. 한 지령을 다시 주거나 이미 실행된 지령들을 보려면 리력목록을 리용할수 있다.

실행된 지령들은 번호로 참조되므로 지령입력재촉문에 번호가 나타나게 하여야 한다. 파일 .cshrc 에 있는 다음의 행에는 체계이름뒤에 지령번호가 있다.

```
set prompt = "$system \ !: "
sys1 1:
```

쉘과 환경변수들을 간단히 고찰하자.현재는 $system 이 체계이름 "sys1"에 대응된다는것을 알면 된다.

사용자는 이미 실행된 지령들중에서 보관하려고 하는 지령들의 개수를 규정할수 있으며 history 지령을 주어 그 지령들을 볼수 있다. .cshrc 에 있는 아래의 행은 리력목록을 20 으로 설정한다.

```
set history = 20
```

지령 history 을 주면 마지막에 실행된 20 개의 지령들이 현시된다. 변수 savehist 는 작업을 끝낸후에 규정된 리력지령들의 개수를 보관한다. 작업을 끝낼 때 리력목록은 지워진다. 이 변수는 값 20 을 가지므로 다음번에 가입할 때 리력목록에는 선행작업프로쎄스에 보관되였을수 있는 20 개 지령들이 있게 된다.

지령행리력

리력목록을 각이한 방법으로 볼수 있다. 우선 마지막 20 개 지령들을 간단히 history 지령을 주어 볼수 있다.

```
sysl 23: history
   4  whoami
   5  pwd
   6  find / -name login -print &
   7  hostname
   8  who
   9  more /etc/passwd
   10  history
   11  history 5
   12  echo $PATH
   13  more .login
   14  cat .login
   15  exit
   16  exit
   17  history -h
   18  pwd
   19  whoami
   20  cd /tmp
   21  cat database.log
   22  cd
   23  history
sysl 24:
```

또한 아래의 실례에서 보여 주는것처럼 리력목록을 행번호가 없이 인쇄할수 있다.

```
sysl 24: history -h
pwd
find / -name login -print &
hostname
who
more /etc/passwd
history
history 5
echo $PATH
```

```
more  .login
cat  .login
exit
exit
history  -h
pwd
whoami
cd  /tmp
cat  database  .log
cd
history
history  -h
sys1  25:
```

그다음 리력목록을 거꿀순서로 인쇄할수 있다.

```
sysl  25: history  -r
    25  history  -r
    24  history  -h
    23  history
    22  cd
    21  cat  database  .log
    20  cd  /tmp
    19  whoami
    18  pwd
    17  history  -h
    16  exit
    15  exit
    14  cat    .login
    13  more    .login
    12  echo  $PATH
    11  history  5
    10  history
    9    more  /etc/passwd
    8    who
    7    hostname
    6    find  /  -name  login  -print  &
sysl  26:
```

또한 리력목록으로부터 인쇄하려고 하는 사건들의 번호를 선택할수 있다. 아래의 실례는 리력목록에 있는 마지막 10개의 지령들을 인쇄한다.

```
sysl 26: history 10
    17 history -h
    18 pwd
    19 whoami
    20 cd /tmp
    21 cat database .log
    22 cd
    23 history
    24 history -h
    25 history -r
    26 history 10
sysl 27:
```

이와 같이 이미 실행된 지령들의 목록을 보는 방법은 다양하다. 표 11-1 에서는 이 항목에서 준 지령들을 개괄하고 있다.

표 11-1 지령행리력

지령	설 명	실례
history	리력목록이 생성되는데 매 지령에 번호가 붙는다.	history
history-h	리력목록은 행번호없이 생성된다.	history-h
history-r	리력목록이 거꿀순서로 생성되는데 매 지령에 번호가 붙는다.	history-r
history *n*	리력목록에 있는 마지막 n 개 지령들이 생성되는데 매 지령에 번호가 붙는다.	history 10

리력목록으로부터 지령들을 재실행

다양한 방법들을 리용하여 리력목록으로부터 지령들을 재실행시킬수 있다.

마지막지령을 !!로, 두번째 지령을 !2 로, "c"로 시작되는 마지막지령을 !c 로 재실행시킬수 있다. 마지막 20 개의 지령들을 얻기 위하여 history 지령을 주고 그다음 그중에서 몇개를 재실행시키자.

```
sysl 27: history
    8  who
    9  more  /etc/passwd
    10  history
    11  history  5
    12  echo  $PATH
    13  more  .login
    14  cat  .login
    15  exit
    16  exit
    17  history  -h
    18  pwd
    19  whoami
    20  cd  /tmp
    21  cat  database  .log
    22  cd
    23  history
    24  history  -h
    25  history  -r
    26  history  10
    27  history
sysl 28:
```

우선 !!로 마지막지령을 다시 준다.

```
sysl 28: !!
history
    9        more  /etc/passwd
    10       history
    11       history  5
    12       echo  $PATH
    13       more  .login
    14       cat  .login
    15       exit
    16       exit
    17       history  -h
    18       pwd
    19       whoami
    20       cd  /tmp
```

```
    21        cat  database  . log
    22        cd
    23        history
    24        history  -h
    25        history  -r
    26        history  10
    27        history
    28        history
sys1  29:
```

다음 !19 로 19 번째 지령을 다시 준다.

```
sys1   29:   !19
whoami
martyp2
sys1   30:
```

이제 !p 로 "p"로 시작되는 마지막지령을 다시 준다.

```
sys1   30:   !p
pwd
/home/martyp2
sys1   31:
```

표 11-2 에서는 흔히 리용되는 리력목록재호출지령들중의 일부를 보여 주고 있다.

표 11-2　　　　　　리력목록으로부터의 재호출

지 령	설　　명	실 례
!N	지령 N 을 실행	**!2**
!!	마지막지령을 실행	**!!**
!-N	마지막으로 실행된 지형으로부터 N 번째 지령을 실행	**!-N**
!str	str 로 시작되는 마지막지령을 실행	**!c**
!?str?	지령행의 그 어디엔가 str 가 들어 있는 마지막지령을 실행	**!?cat?**
!{str1}str2	str1 이 있는 마지막지령에 str2 를 첨가	**!{cd}/tmp**
^str1^str2^	마지막지령에서 str1 을 str2 로 치환	**^cat^more^**

.cshrc 에서의 별명

별명(alias)은 자주 리용되는 지령 또는 지령렬에 대하여 선택되는 이름이다. 많은 별명들이 이미 정의되여 있다.

.cshrc 파일을 별명이 보관되는 장소로 리용하고 체계에 가입할 때마다 읽을수 있다. 또한 지령행입력재촉문에서 별명을 정의할수 있으나 그것들은 작업을 끝낼 때 지워 진다.

아무런 인수도 없이 alias 지령을 주면 모든 별명들이 렬거된다. 이 목록에는 이미 설정된 별명들뿐아니라 사용자가 설정한 별명들도 포함된다. 아래의 지령은 현재 작업하고 있는 체계상에 미리 설정되여 있는 별명들을 보여 준다.

```
sys1  7:  alias
d         dirs
m         more
pd        pushd
pd2       (pushd  +2)
po        popd
status    (date;  bdf)
sys1  8:
```

이미 설정된 별명들을 리용하는데는 제한이 없다. 사용자자신의 별명을 창조하려면 우선 alias 지령을 주고 별명의 이름뒤에 그 별명이 실행될 때 실행될 지령들을 지적해 주면 된다.

이제 간단한 별명들을 창조해 보자. 첫번째 행은 history 지령에 대하여 "h"라는 별명을 창조한다.

```
sys1  1:  alias  h  history
sys1  2:  h
        history
```

h 를 입력할 때마다 hisotory 지령이 실행된다.

아래의 실례에서는 공백을 포함하는 지령의 별명을 창조한다. 공백을 포함하는 지령은 웃반점안에 포함시킨다.

```
alias  ls='ls  -al'
alias
ls
```

첫 지령은 지령 ls ?al 에 대한 별명 "ls"를 창조한다. 그다음에 새 별명이 실지로 별

명들의 목록에 나타나는가를 보기 위하여 alias 지령을 주었다. 다음으로 지령 ls ?al 이 실행되는가를 보기 위하여 ls 를 실행시켰다.

별명을 설정한 다음 그 별명을 작업의 전 과정에 유지하지 않아도 된다. 어느 한 별명이 마음에 들지 않으면 unalias 지령을 리용하여 그 별명을 제거할수 있다.

unalias 의 작용을 보기 위하여 다시 별명들의 목록을 생성하고 별명 h 를 없애기 위하여 unalias 를 리용하며 그것이 실지로 제거되였는가를 보기 위하여 별명 h 를 실행시킨다.

```
unalias   h
h
```

unalias 를 실행시켜 별명 h 를 제거한 다음에 별명 h 를 실행시키려고 할 때 h 를 찾을수 없다는 통보문이 표시된다.

sys1 3: **alias procs 'echo "Number of processes are: \c"; ps -ef | wc -l'**
                    # single quote on outside
                    # double quote on inside

procs 를 실행시키면 다음과 같은 통보문이 표시된다.

sys1 4: **procs**
Number of processes are: 44

표 11-3 을 보면 이 지령행에 들어 있는 인용문을 리해할수 있다.

표 11-3 쉘인용문

기호들	설 명
'cmd'	웃반점은 기호렬이 문자그대로 취급된다는것을 의미한다.
"str"	웃두점은 지령 및 변수치환을 허락한다는것을 의미한다.
\c	esc 기호인데 인쇄할 때 그뒤의 기호렬이 행바꾸기없이 인쇄되도록 한다.
'str'	이것은 지령을 실행하고 출력을 치환한다는것을 의미한다.

표 11-3 을 별명 "procs"에 적용하면 이 별명이 무엇을 포함하는가를 알수 있다. 그 별명은 웃반점으로 시작된다. 웃반점안에 들어 있는것은 지령을 의미한다. 첫 지령은 echo 지령인데 이 지령은 출력할 기호렬을 웃두점안에 넣어 리용한다. 인쇄될 때 행바꾸기를 금지시키는 기호 \c 를 첨가할수 있다. 두반점(;)은 지령들을 분리시킨다. ps 가 실행되면 프로세스들의 목록이 wc 에로 출력되여 프로세스들의 개수가 얻어 진다. wc 에

의하여 현재 43개의 프로세스들이 실행되고 있다는것을 알수 있다.

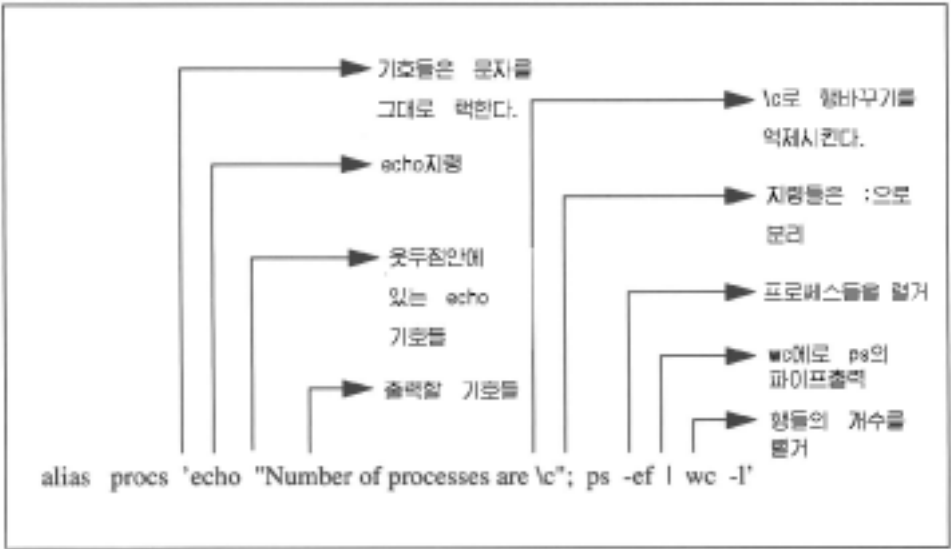



그림 11-3. 인용문실례

그림 11-3에서 볼수 있는바와 같이 일부 인용문들은 리해하기가 어렵다. 쉘스크립트들을 변경하거나 다시 리용하려면 인용문을 리해하는것이 중요하다.

파일이름확장

C쉘을 리용하는데서 또 다른 중요한 부분은 파일이름확장이다. 파일이름들을 취급하는 쉘스크립트들을 작성하기전에 파일이름확장에 대하여 아는것이 중요하다. 표 11-4에서는 몇가지 공통적인 파일이름확장과 패턴정합을 보여 주고 있다.

표 11-4　　　　파일이름확장과 패턴정합

기호들	실 례	설 명
*	1) ls*.c	령 또는 그이상 기호들을 맞춘다.
?	2) ls conf.?	임의의 한개 기호를 맞춘다.
[list]	3) ls conf.[co]	목록에 있는 임의의 기호를 맞춘다.
[lower-upper]	4) ls libdd.9873[5-6].sl	범위에 있는 임의의 기호를 맞춘다.
str{str1, str2, str3, …}	5) ls ux*.{700, 300}	str를 { }의 내용으로 확장한다.
~	6) ls -a~	홈등록부이다.

표 11-4에 있는 실례들을 자세히 서술하면 다음과 같다.

① 등록부에서 ".c"로 끝나는 모든 파일들을 렬거하려면 다음과 같이 한다.

sys1 30: **ls *.c**

conf.SAM.c conf.c

② 등록부에서 "conf"로 이름 지어 져 있고 한개 기호로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

sys1 31: **ls conf.?**

conf.c conf.o conf.1

③ 등록부에서 "conf"로 이름 지어 져 있고 한개 기호 "c" 또는 "o"로 된 확장자를 가진 모든 파일들을 렬거하려면 다음과 같이 한다.

sys1 32: **ls conf.{co}**

conf.c conf.o

④ 류사한 이름들을 가지면서 어떤 범위의 마당들을 가지는 파일들을 렬거하려면 다음과 같이 한다.

sys1 46: **ls libdd9873[5-6].sl**

libdd98735.sl libdd98736.sl

⑤ "ux"로 시작되고 확장자가 "300" 또는 "700"인 파일들을 렬거하려면 다음과 같이 한다.

sys1 59: **ls ux*.{700, 300}**

uxbootlf.700 uxinstfs.300 unistkern.300
unistkern.700 unistlf.700

⑥ 홈등록부에 있는 파일들을 렬거하려면 아래에서 보여 주는것처럼 "～"를 리용한다.

sys1 62: **ls -a~**

.	.cshrc.org	.login	.shrc.org
..	.exrc	.login.org	.vue
.chsrc	.history	.profile	.vueprofile

⑦ 사용자의 홈등록부에 있는 파일들을 렬거하려면 다음과 같이 한다.

sys 1 65: **ls -a ~gene**

.	.history	.vue	splinedat
. .	.login	.vueprofile	trail.txt
.chsrc	.login.org	ESP-File	under.des
.cshrc.org	.profile	Mail	xtra.part
.exrc	.shrc.org	opt	

이 방법들은 쉘스크립트를 작성할 때 리용되므로 파일이름확장에 익숙되여야 한다.

방향바꾸기(I/O 방향바꾸기)

UNIX 는 지령들의 입력을 건반(표준입구장치)으로부터 받으며 출력을 화면(표준출구장치)에 보내도록 설치되여 있다. 지령들은 오유정보도 화면에 보낸다. 이 표준설정을 무시하고 입력이 다른 곳으로부터 들어 오고 출력과 오유들이 다른 곳으로 나가게 할수도 있다. 이것을 방향바꾸기라고 부른다. 표 11-5 에서는 방향바꾸기의 형식들을 보여 주고 있다.

표에서 보는바와 같이 지령의 출력을 표준출구장치로부터 파일에로 바꾸려면 ">"을 리용하여야 한다. noclobber 라고 부르는 환경변수를 설정하면 이미 존재하는 파일에로의 방향바꾸기는 진행되지 않는다. 실례로 다음의 /tmp/processes 와 같은 이미 존재하는 파일에 기입하려고 시도한다면 그 파일이 존재한다는 통보문을 받는다.

ps -ef > /tmp/processes
/tmp/processes: File exists

그러나 파일을 덧쓰도록 방향바꾸기에 "!"를 리용할수 있다. ">!"를 리용하면 파일은 덧씌여 진다. 그리고 ">>!"은 이미 존재하는 파일의 뒤에 출력이 첨가되도록 한다. 이에 대한 실례들을 표 11-5 에서 보여 주고 있다.

표 11-5 흔히 리용되는 방향전환형식들

지령 또는 대입	실 례	설 명
<	wc -l < .login	표준입력방향전환:wc(단어계수)를 실행하고 .login 에 있는 행들의 수를 렬거한다.
>	ps -ef > /tmp/processes	표준출력방향전환:ps 를 실행하고 출력을 파일 /tmp/process 에로 보낸다.
>>	ps -ef >> /tmp/processes	표준출력을 첨가:ps 를 실행하고 출력을 파일 /tmp/process 에 첨가
>!	ps -ef > ! /tmp/processes	출력방향전환을 첨가하고 noclobber 를 무시한다. /tmp /processes 가 존재하면 그우에 덧쓴다.
>>!	ps -ef >>! /tmp/processes	표준출력을 첨가하고 noclobber 를 무시한다. /tmp/processes 의 끝에 첨가한다.
| (pipe)	ps | wc -l	ps 를 실행하고 그 결과를 wc 의 입력으로 리용한다.

(표계속)

지령 또는 대입	실 례	설 명
0 - standard input		
1 - standard output		
2 - standard error	cat program 2> errors	파일 program 을 표준출력에 cat 하고 오유들을 파일 errors 에로 방향전환한다.
	cat program 2>> errors	파일 program 을 표준출력에 cat 하고 오유들을 파일 errors 에로 첨가한다.
	cat program>&outfile	파일 program 과 오유들을 파일 outfile 에 cat 한다.
	(cat program>outfile) >&errors	파일 program 을 outfile 에 cat 하고 오유들을 파일 errors 에 방향전환한다.

쉘과 환경변수

C 쉘에서 사용자환경에 대한 정보는 쉘변수들과 환경변수들에 보관되여 있다.

쉘변수들을 때때로 **국부변수**(Local Variable)들이라고 부른다. 쉘변수들은 그 변수들이 창조된 쉘에만 알려 져 있다.

환경변수들은 때때로 **전역변수**(Global Variable)들이라고 부른다. 환경변수들은 그 변수들이 창조된 쉘에서 정의되며 원형쉘로부터 파생된 모든 쉘에 전달된다. 이 변수들은 원형쉘로부터 파생된 쉘들에 전달되기때문에 그것들은 대역적으로 고찰된다.

아래의 실례에서 보여 주는바와 같이 쉘변수들을 set 지령으로, 환경변수들을 env 지령으로 볼수 있다.

```
sys1 22: set
argv    ( )
autologout          600
cwd         /home/martyp2
history     20
home        /home/martyp2
noclobber
path        (/usr/bin  /usr/ccs/bin  /usr/contrib/bin  /opt/nettladm/bin  /opt/pd/bin/)
prompt      sys1 ! :
savehist    20
shell       /usr/bin/csh
```

```
status        0
system        sys1
term          vt100
sys1 23： set
argv          ( )
autologout          600
cwd           /home/martyp2
history       20
home          /home/martyp2
noclobber
path          (/usr/bin  /usr/ccs/bin  /usr/contrib/bin  /opt/nettladm/bin  /opt/pd/bin/)
prompt        sys1 ! ：
savehist            20
shell         /usr/bin/csh
status        0
system        sys1
term          vt100
sys1 24：

sys1 24： env
HOME=/home/martyp2
PATH=/usr/bin：/usr/ccs/bin：/usr/contrib/bin：/opt/nettladm/bin：/opt/pd/bin：/opt/n
LOGNAME=martyp2
TERM=vt100
SHELL=/usr/bin/csh
MAIL=/var/mail/martyp2
COLUMNS=80
LINES=24
MANPATH=/usr/share/man/%L：/usr/share/man：/usr/contrib/ man/%L：/usr/contrib/man：/n
TZ=PST8PDT
sys1 25：
```

국부변수들을 작은 글자로, 전역변수들을 큰 글자로 표시한다.

set 지령을 실행시키면 많은 국부 및 전역변수들이 설정된것을 볼수 있다. 변수가 설정되였는가를 특수한 표식 $?를 리용하여 결정할수 있다. 이 특수한 표식을 변수이름의 앞에 놓으면 그 변수가 설정된 경우에는 1 을 주고 설정되지 않은 경우에는 0 을 준다. 아래의 실례에서는 변수 history 와 hist 가 설정되였는가를 보기 위하여 echo 지령과 이 특수한 표식을 리용한것을 보여 주고 있다.

```
sys1  25:  echo  $?history
1
sys1  26:  echo  $?hist
0
sys1  27:
```

변수 history 로부터 1 을 주었기때문에 이 변수는 설정되였다는것을 알수 있다. 이것을 앞에서 본 set 지령의 출력으로 확인하면 history 는 20 으로 설정되여 있다.

변수 hist 로부터는 0 이 주어 졌기때문에 그 변수는 설정되지 않았다는것을 알수 있다.

쉘은 특수한 기호 $를 리용하여 변수의 값을 얻어 낼수 있다. 아래의 실례에서는 변수 history 의 값을 얻기 위하여 $를 앞에 붙여 echo 지령에 결합한것을 보여 주고 있다.

```
sys1  27:  echo  $history
20
sys1  28:  echo  $hist
hist:  Undefined  variable.
sys1  29:
```

쉘변수들은 set 를 리용하여 정의된다. 앞에서 본 cshrc 파일에서는 쉘변수 hisotory 가 다음의 지령으로 설정되였다.

```
set  history=20
```

환경변수들은 아래의 지령에서와 같이 setenv 로 정의된다.

```
setenv  EDITOR  vi
```

배경일감과 일감조종

지금까지 본 많은 실례들에서처럼 지령을 실행시키면 그 지령이 완성될 때까지 입력재촉문이 나타나지 않는다. 일부 지령들은 완성되는데 많은 시간을 소비하므로 입력재촉문이 다시 나타날 때까지 오래동안 기다려야 한다. 입력재촉문이 다시 나타나기를 기다리면서 그 지령을 배경에서 실행시킬수 있다. UNIX 는 다중과제체계이기때문에 많은 지령들을 배경에서 실행시키고 다른 지령들을 더 줄수 있도록 입력재촉문을 제공해준다.

지령을 배경에서 실행시키려면 간단히 지령행의 끝에 &를 붙여 주어야 한다. 지령행의 끝에 한개의 &를 붙여 주면 그 지령은 배경에서 실행되며 입력재촉문은 곧 다시 나타난다.

이제 몇가지 지령들을 Solaris 체계에서 실행시켜 보자. 우선 ".c"로 끝나며 완결되는

데 일정한 시간이 걸리는 모든 파일들을 /usr 에서 찾기 위한 지령을 실행시키자. 기호렬 find 를 time 지령과 함께 리용하여 그 지령이 완결되는데 얼마나 오랜 시간이 걸리는가를 보기로 하자.

```
martyp $ time find /usr -name *.c
          ⋮
/usr/demo/link_audit/src/dumpbind.c
/usr/demo/link_audit/src/env.c
/usr/demo/link_audit/src/hash.c
/usr/demo/link_audit/src/perfcnt.c
/usr/demo/link_audit/src/symbindrep.c
/usr/demo/link_audit/src/truss.c
/usr/demo/link_audit/src/who.c
find: cannot read dir /usr/aset: Permission denied

real      1m21.04s
user      0m1.51s
sys       0m12.67s
```

이 지령이 완결되는데 대략 1min 21s 걸렸다. 이 지령이 전경에서 실행되였기때문에 입력재촉문이 되돌려 질 때까지 기다려야 하였으며 그동안 다른 지령을 줄수 없었다.

지령을 배경에서 실행시키고 입력재촉문을 즉시 되돌려 받기 위하여서는 아래의 실례에서 보여 주는바와 같이 지령을 줄 때 뒤에 &를 붙여 주어야 한다.

```
martyp $ time find /usr -name *.c > cprogs 2>&l &
[3]       16279
martyp $
real      2m10.20s
user      0m1.31s
sys       0m8.62s
```

이 지령을 배경에서 실행시킨 결과는 첫째로, 각괄호안에 넣은 일감번호이며 둘째로, 프로쎄스 id(PID)이다. 오유들을 포함하여 이 지령의 모든 출력들은 파일 cprogs 에 기입된다. 입력재촉문은 지령을 준 다음 즉시 되돌려 지며 그 지령이 완결된후 time 의 출력이 화면에 현시된다. 배경에서 실행되도록 지령을 준 다음에는 즉시 다른 지령을 주기 시작할수 있다.

사용자는 **전경일감**(Foreground Job)과 **배경일감**(Background Job)들을 조종할수 있다. 전경일감을 정지시키려면 아래의 실례에서 보여 준바와 같이 ^z(crtl-z)를 누르면 된다.

```
martyp $ find /usr -name *.c
/usr/openwin/share/include/x11/Xaw/Template.c
/usr/openwin/share/src/dig_samples/DnD/main.c
/usr/openwin/share/src/dig_samples/DnD/owner.c
/usr/openwin/share/src/dig_samples/DnD/requestor.c
/usr/openwin/share/src/dig_samples/Tooltalk/olit_tt.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_callbacks.c
/usr/openwin/share/src/dig_samples/Tooltalk/tt_code.c
/usr/openwin/share/src/dig_samples/cel/ce_mapi.c
/usr/openwin/share/src/dig_samples/ce2/ce_simple.c
/usr/openwin/share/src/dig_samples/dnd_olit/olitdnd.c
/usr/openwin/share/src/dig_samples/dnd_xview1/xview_dnd.c
/usr/openwin/share/src/dig_samples/dnd_xview2/xview_dnd2.c
/usr/openwin/share/src/dig_samples/selection_olit/olit_sel.c
/usr/openwin/share/src/dig_samples/tooltalk_simple/tt_send.c
/usr/openwin/share/src/olit/oldials/oldials.c
/usr/openwin/share/src/olit/olitbook/ch10/draw.c
^Z[3] + Stopped (SIGTSTP)        find /usr -name *.c
```

ctrl-z 를 누르면 find 지령은 정지되며 이때 사용자는 일감번호(이 경우에 3)를 볼수 있고 그 일감의 상태는 "Stopped"로 표시된다. 이 지령은 완전히 끝난것이 아니라 일시 정지되였을뿐이다. 이 프로쎄스를 fg 지령으로 전경에서 실행시키거나 bg 지령으로 배경에서 실행시킬수 있다. bg 지령은 지령의 뒤에 &를 붙여 준것과 같다. 이 지령은 중단된 위치에서 시작된다. 지령 fg 또는 bg 을 줄 때 일감번호를 지적하면 안된다. 왜냐하면 표준적으로 마지막일감(이 경우에 일감번호 3)에서 규정된 조작을 실행하기때문이다.

이 실례에서는 일감번호 3 을 정지시켰는데 이것은 보다 작은 번호를 가지고 실행되는 다른 일감들이 있다는것을 의미한다. jobs 지령을 리용하여 모든 일감들의 목록과 그것들의 상태를 얻을수 있다. 또한 지령이 전경에서 실행되도록 fg 에 %와 일감번호를 붙여 주거나 지령이 배경에서 실행되도록 bg 에 %와 일감번호를 붙여 주는 방법으로 일감들을 조종할수 있다. 일감을 종결시키려면 kill 지령에 %와 일감번호를 붙여 주면 된다.

아래의 실례에서는 지령 jobs 로 모든 지령들을 렬거하는것과 kill 지령으로 일감 1 과 일감 3 를 제거하며 일감 3 을 배경에서 실행시키는것을 보여 주고 있다.

```
martyp $ jobs
[3] + Stopped (SIGTSTP) find /usr -name *.c
[2] - Running        time find / -name gnu* > gnu 2>&1 &
[1] Running                 time find / -name *.c > cprogs 2>&1 &
martyp $ kill %1
[1] Terminated time find    -name *.c > cprogs 2>&1 &
```

martyp $ **kill %2**
[2] - Terminated   time find / -name gnu* > gnu 2>&1 &
martyp $ **bg %3**
[3] find /usr -name *. c&
martyp $

일감 3 이 배경에서 실행되도록 &를 붙여 주었다. 일감 1 과 일감 2 를 제거하고 일감 3 을 배경에서 실행시키면 입력재촉문을 즉시 돌려 주므로 사용자는 다른 일을 수행할수 있다.

umask 와 허락

csh 와 관련하여 취급할 추가적인 문제는 **파일허락**(File Permission)들과 그것들이 umask 에 관련되는 방식이다. 이것은 모든 사람들이 리용할수 있는 쉘프로그람들과 제한된 수의 사용자들(가능하게는 체계관리자)만이 리용할수 있는 쉘프로그람들을 작성하려고 하는 경우에 중요한 문제로 제기된다. umask 는 새 파일들과 등록부들에 대한 허락을 규정하는데 리용된다.

다음의 실례를 고찰하자. 이 실례에서는 ls -l 을 리용하고 있다.

sys1 1: **ls -l script1**
-rwxr-xr-x 1 marty users 120 Jul 26 10:20 script1

이 파일에 대한 **접근권한**(Access Right)은 지령 ll 을 줄 때 읽기(r), 쓰기(w), 실행(x)의 위치에 의하여 정의된다. 그림 11-4 에서는 이 파일에 대한 3 가지 접근권한들의 그룹들을 보여 주고 있다.

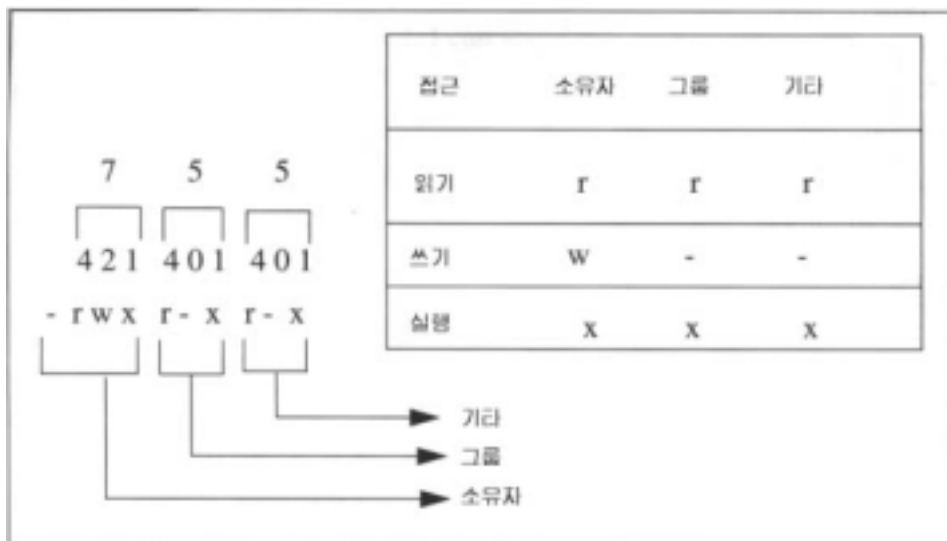



그림 11-4. 파일허락의 실례

파일소유자는 자기의 파일에 대한 읽기, 쓰기, 실행허락을 가진다. 사용자가 속하는 그룹과 기타 다른 사람들은 읽기와 실행허락을 가진다. 파일에 대한 허락들은 매 마당의 8 진합에 의하여 규정된다. 이 경우에는 755 이다.

새로운 쉘스크립트 또는 임의의 새로운 파일을 만들면 무슨 문제가 제기되는가, 어떤 허락설정들이 존재하는가 하는 문제가 제기된다. 쉘스크립트를 실행시키려면 이 파일에 대한 실행허락이 필요된다. umask 를 리용하여 모든 새로운 파일들과 등록부들에 대한 표준값을 정의할수 있다.

기정으로 대부분의 체계들은 등록부에 대하여서는 777, 파일들에 대하여서는 666 의 허락을 가지고 시작한다. 이것은 누구나 다 모든 등록부들에 대하여 완전한 접근을 가지며 파일들에 대하여서는 읽기와 쓰기접근을 가진다는것을 의미한다. 이 표준값들은 umask 의 값으로 변경된다.

다음과 같은 두가지 방법으로 umask 를 볼수 있다.

```
martyp $ umask
002
martyp $ umask -S
u=rwx, g=rwx, o=rx
martyp $
```

첫번째 실레에서는 umask 의 8 진값을 현시하며 두번째 실레에서는 umask 의 기호적인 값을 보여 준다. -S 는 모든 UNIX 변종들에서 동작하지 않는다. umask 는 접근을 불가능하게 하며 3 개의 8 진마당들을 리용한다. 그 마당들은 그림 11-5 에서 보여 주는바와 같이 사용자와 그룹들 그리고 기타 다른 사람들에 대한 접근코드들의 합이다.

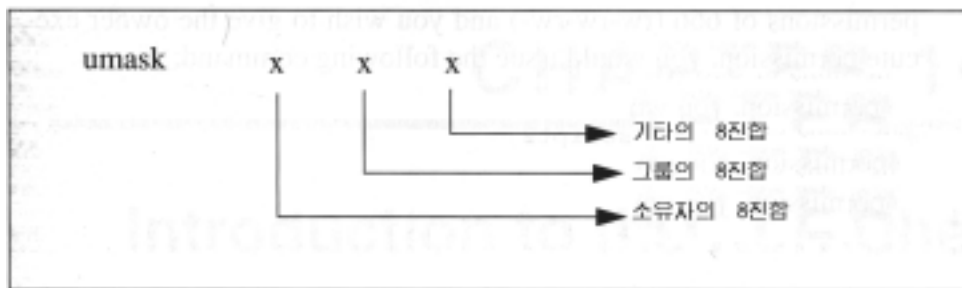



그림 11-5. umask 마당들

umask 마당의 부정은 표준설정과 론리적되여 umask 를 변경시킨다. umask 의 값을 설정해 줄수 있다. 앞의 실레에서 우리는 umask 를 두가지 방법으로 보았다. umask 를 설정하려면 간단히 umask 지령과 요구되는 값을 주면 된다. 실례로 umask 를 022 에 설정하면 그림 11-6 에서 보여 주는것처럼 "group"과 "other"에 대하여 등록부의 쓰기허락이 제거된다.

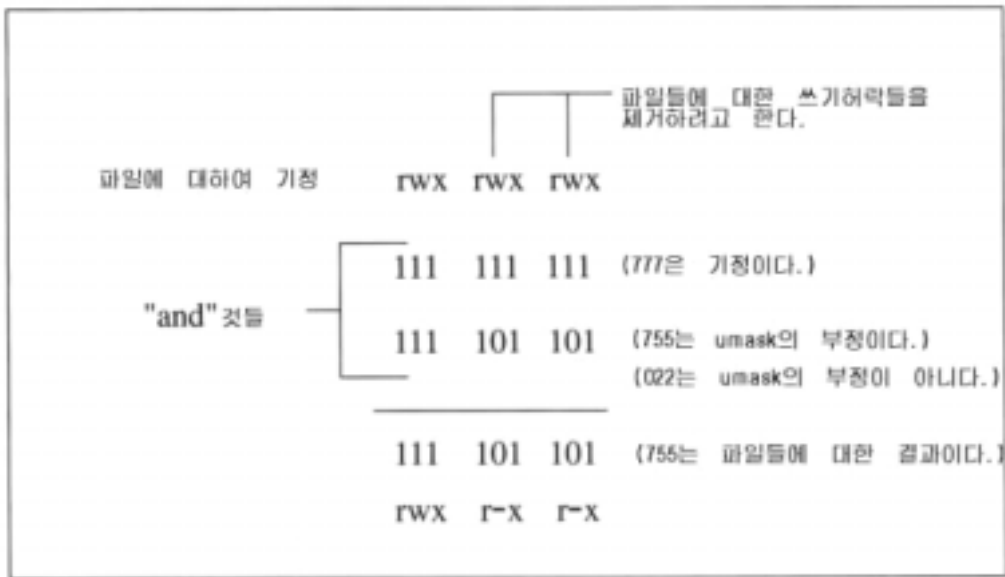



그림 11-6. umask 실례

이 실례에서 umask 022는 파일허락을 755로 변경시킨다.

류사하게 022인 umask는 파일들에 대하여 666인 허락을 644로 변경시킨다. 이것은 그룹과 기타에 대하여 읽기허락만을 의미한다.

chmod로 파일허락의 변경

chmod 지령은 파일에 대한 허락을 변경시키는데 리용된다. 우에서 서술된바와 같이 umask로 무엇을 할수 있는가에는 상관없이 chmod로 임의의 시각에 파일들에 대한 허락을 변경시킬수 있다. 대부분의 경우들에 chmod로 파일허락을 변경시키는것은 그 파일의 소유자 또는 상급사용자만이 할수 있다. chmod에 대한 론의를 sort를 가지고 시작하자.

```
$ ls -l sort
-rwxr-x--x   1 marty users 120 Jul 26 10:20 sort
```

그림 11-7에서는 sort에 대한 허락의 분해를 보여 주고 있다.

사용자들은 정의되는 파일형에 대하여 조종을 많이 할수 없다. 그러나 파일이 사용자의것일 때에는 그 파일의 허락을 마음대로 조종할수 있다. chmod 지령은 파일 또는 등록부에 대한 허락을 변경시키는데 리용된다. 만일 사용자가 그 파일의 소유자이면 사용자는 그 파일에 대한 허락을 변경시키는 마당만을 가질수 있다.

허락을 변경시킬수 있는 방법은 두가지 즉 기호적인것과 수값적인것이 있다. 수들을 관리하는것이 더 쉬우므로 우선 수값방식을 취급하고 그다음에 기호들을 취급하며 chmod 개요에 기호들의 의미들을 포함시킨다.

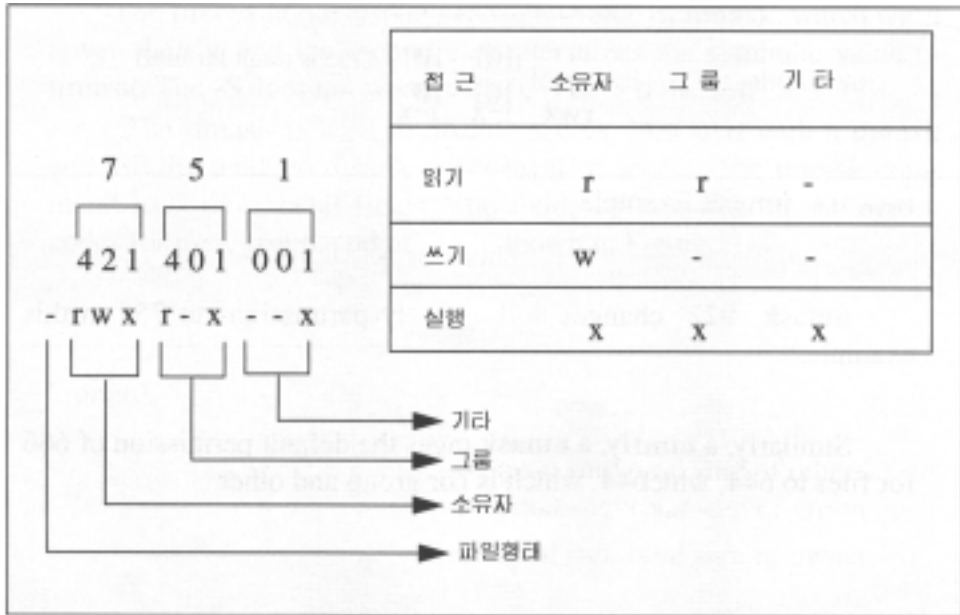



그림 11−7. 파일 sort 의 허락

수라는것은 무엇을 의미하는가? sort 에 대한 수들을 보면 751 이라는 허락이 있다. 7 은 소유자(100 의 자리), 5 는 그룹(10 의 자리), 1 은 기타(1 의 자리)에 대한것이다. 그림 11−8 에서는 위치들의 의미를 설명해 주고 있다.

owner, group, other 에 대하여 요구되는 허락을 선택하려면 chmod 지령을 리용하여 그 허락들을 파일 또는 등록부에 할당한다. 이 허락가능성들중의 일부(실례로 실행허락)는 자주 리용되지 않는다. 왜냐하면 파일을 실행시키기 위해서는 그 파일이 읽기허락으로 되여 있어야 하기때문이다. 그러나 그림 11−8 에서는 모든 가능성들을 다 포함시켰다. 그림 11−8 에서 보여 준 허락방식비트들외에도 현재는 필요되지 않는 기타 방식비트들이 있다.

만일 group 에 대하여 sort 의 쓰기허락을 첨가하고 other 에 대하여 모든 허락들을 제거하려면 chmod 지령을 줄 때 알맞는 수값을 함께 주면 된다. 아래의 지령모임은 우선 sort 에 대하여 이미 존재하는 허락들을 렬거하고 변경시키며 마지막으로 새로운 허락을 렬거한다.

```
$ ls -l sort
-rwxr-x--x    1    marty    users    120    Jul    26    10:20    sort
$ chmod 770 sort
$ ls -l sort
-rwxrwx---    1    marty    users    120    Jul    26    10:20    sort
```

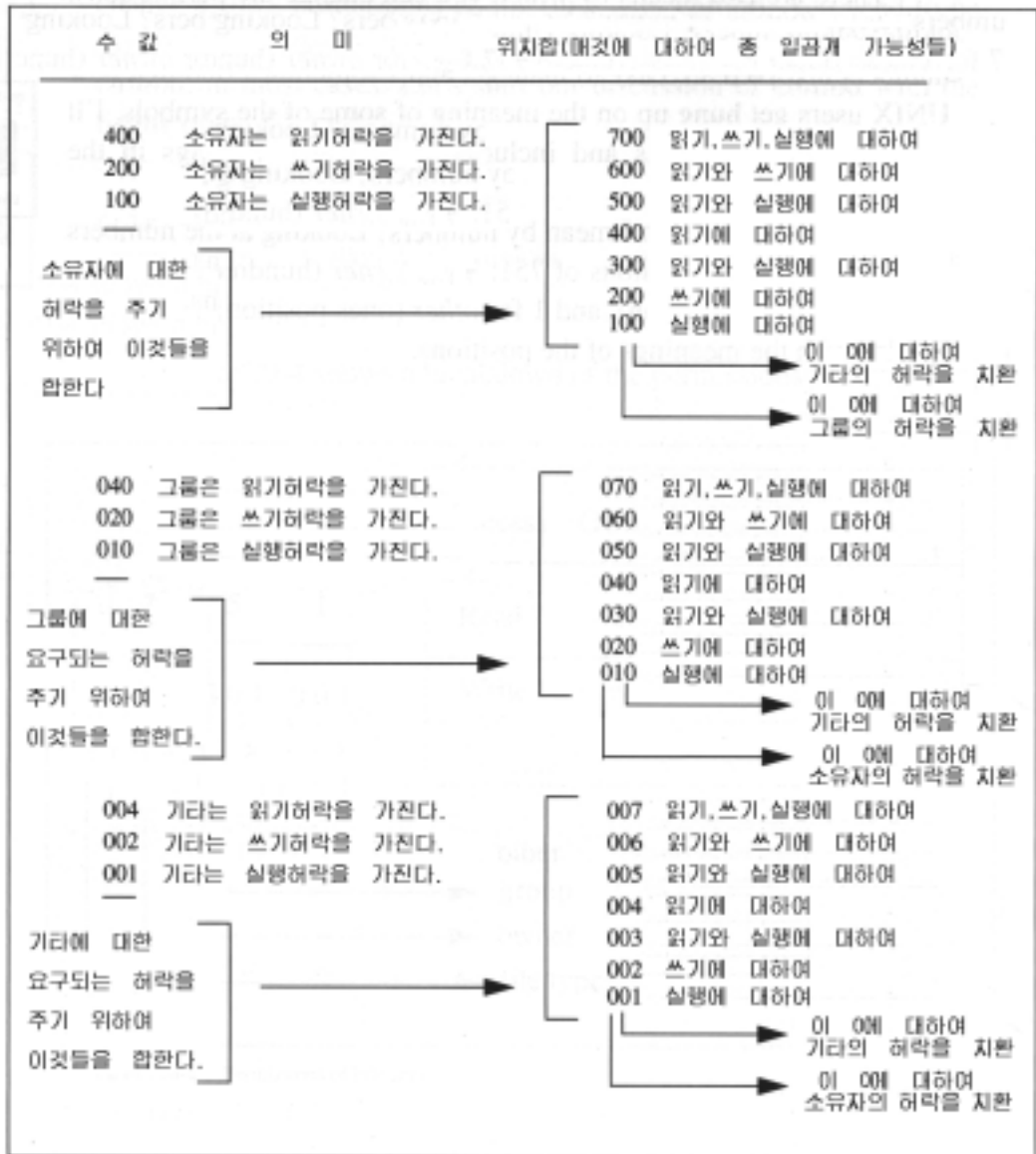



그림 11-8. 수값허락개괄

기호방식을 리용하여 허락들을 변경시키는 동일한 지령모임은 실례로 다음과 같다.

```
$ ls -l sort
-rwxr-x--x   1   marty   users   120   Jul   26   10:20   sort
$ chmod g+w, o-x sort
$ ls -l sort
-rwxrwx---   1   marty   users   120   Jul   26   10:20   sort
```

기호방식에서는 chmod 지령을 주고 변경에 의하여 영향을 받을 사람(사용자(u), 그룹(g), 기타(o), 모두(a)), 수행하려고 하는 연산(첨가(+), 삭제(-), 교체(=)), 규정하려고 하는 허락(읽기(r), 쓰기(w), 실행(x))을 규정해 준다. 기호방식을 리용하는 선행실례에서는 그룹(g)에 대하여 쓰기(w)허락이 첨가(+)되고 기타(o)에 대하여 실행(x)허락이 제거(-)되였다.

아래에 chmod에서 흔히 리용되는 기호들을 개괄하였다.

영향을 받는 사람의 기호:

u	사용자가 영향을 받는다.
g	그룹이 영향을 받는다.
o	기타가 영향을 받는다.
a	모든 사용자들이 영향을 받는다.

수행할 조작:

+	허락을 첨가
-	허락을 제거
=	허락을 교체

규정된 허락:

r	읽기허락
w	쓰기허락
x	실행허락
u	사용자허락을 복사
g	그룹허락을 복사
o	기타허락을 복사

지령소개

여기서는 csh에 대한 지령들을 보여 주고 있다. 지령들은 UNIX 변종들에서 차이나므로 일부 지령들에 대한 선택항목들과 다른 령역들에서 차이나는 점들이 있을수 있다. 그러나 다음의 지령들은 좋은 기준으로 리용될수 있다.

csh

csh - C쉘

---

csh(1)

이름

csh − C의 문장형식을 가진 쉘(지령해석기)

형식

csh [-cefinstvxTVX] [command_file] [argument_list …]

해설

csh는 지령리력완충기, C형의 문장론, 일감조종특성들을 통합한 지령언어해석기이다.

지령선택항목들

지령선택항목들은 다음과 같이 해석된다.

-c 존재하여야 하는 다음 인수로부터 지령들을 읽는다. 나머지 인수들은 argv에 배치된다.

-e 호출된 지령이 비정상적으로 종결되거나 령이 아닌 탈퇴상태를 산출하면 C쉘은 탈퇴한다.

-f 홈등록부에 있는 .cshrc파일의 실행을 억제시킨다. 따라서 쉘시동시간이 소비되게 된다.

-i csh가 콤퓨터말단이 아닌 다른 장치로부터 호출되였을 때 호상작용하면서 응답하도록 한다. csh는 표준적으로 호상작용하지 않으면서 응답한다. csh가 콤퓨터말단으로부터 호출되였으면 그것은 항상 어느 선택항목이 선택되였는가를 고려함이 없이 호상작용하면서 응답한다.

-n 문법검사를 하지만 지령들을 수행하지 않는다. 이것은 쉘스크립트들에서 문장론검사를 위하여 리용된다. 리력, 지령, 별명 등 모든 치환들이 수행된다.

-s 표준입력으로부터 지령입력을 받는다.

-t 입력의 한개 행을 읽고 실행시킨다.

-v 리력치환이 진행된후에 지령입력을 표준출력에 내보내게 하면서 쉘변수들을 설정한다.

-x 수행시키기전에 모든 지령들을 표준오유흐름에 출력하면서 echo 쉘변수를 설정한다.

-T 지령/파일이름을 완성하는데 ESC건을 리용하며 이미 존재하는 파일들을 렬거하는데 ctrl-d를 리용하는 tenex 특성을 고찰할수 없게 한다.

-V .cshrc가 실행되기전에 verbose변수를 설정하여 모든 .cshrc지령들이 표준출력에 보내지도록 한다.

-X .cshrc 가 실행되기전에 echo 변수를 설정하여 모든 .cshrc 지령들이 표준출력에 보내지도록 한다.

지령선택항목들을 처리한후에 인수목록에 남아 있는 선택항목 -c, -i, -s, -t 들이 규정되여 있지 않으면 남아 있는 첫 인수가 실행되여야 할 지령파일의 이름으로 취해 진다.

지령들

단순지령은 단어들의 렬인데 첫 단어는 실행되여야 할 지령을 규정한다. 막대기(|)에 의하여 분리되는 단순지령들의 렬은 파이프선을 형성한다. 파이프선에 있는 매 지령의 출력은 파이프선에 있는 다음번 지령의 입력으로 된다. 파이프선들의 렬은 두반점(;)에 의하여 분리되는데 이것은 파이프선들이 순차적으로 실행되도록 한다. 파이프선들의 렬은 마지막항목뒤에 &를 붙여 주면 배경방식에서 실행될수 있다.
파이프선이 괄호안에 놓이면 순서대로 다음번 파이프선의 성분으로 될수 있는 단순지령을 형성한다. 파이프선들은 || 또는 &&에 의하여서도 분리될수 있는데 이것은 C 언어에서처럼 첫번째 파이프선이 실패하거나 성공할 때 두번째 파이프선이 실행되도록 한다.

일감들

csh 는 매 파이프선을 하나의 일감으로 하며 현재 일감들의 표를 유지하고 그 일감들에 작은 옹근수들을 할당한다. 한 일감이 &를 리용하여 비동기적으로 시동될 때 쉘은 다음과 같은 행을 인쇄한다.

[1] 1234

이것은 비동기적으로 시동된 일감이 일감번호 1 이며 프로쎄스 id 가 1234 인 한개 프로쎄스를 가진다는것을 가리킨다. 어느 한 일감이 실행될 때 다른 일을 하려면 정의된 정지기호를 입력하여 그 일감에 정지신호를 보내야 한다(termio(7)을 참고). 그러면 csh 는 표준적으로 그 일감이 정지되였다는것을 가리키고 다른 입력재촉문을 인쇄한다. 그다음 사용자는 이 일감의 상태를 변경시킬수 있다. 즉 지령 bg 로 그 일감을 배경에 놓고 다른 지령들을 실행시키며 다시 지령 bg 로 그 일감을 전경에 놓을수 있다. 지연된 정지기호라는것이 있는데 이것은 프로그람이 그 기호을 읽을 때까지 정지신호를 발생하지 않는다. 이 기호는 정지시키려고 하는 일감에 대한 몇가지 지령들을 준비하기전에 입력될수 있다.

배경에서 실행되고 있는 일감은 말단으로부터 읽으려고 할 때 정지된다. 배경일감들은 표준적으로 출력을 생성할수 있으나 이것은 지령 stty tostop 을 주어 불가능하게 할수 있다(stty(1)을 참고). 선택항목 tty 를 설정하면 배경일감들은 입력을 읽으려고 할 때처럼 출력을 생성하려고 할 때 정지된다.
말단대면부로부터의 건반신호들과 행정지신호들은 체계상의 배경일감들에로 전송되지 않는다. 이것은 배경일감들이 작업끝내기 또는 중단, 탈퇴, 정지, 지연된 정지신호가 입력되는데 영향을 주지 않는다는것을 의미한다(termio(7)을

참고).

쉘에서 일감들을 지적하는 방법은 여러가지이다. 기호 %는 일감이름을 안내하는 기호이다. 실례로 일감번호 1 을 %1 로 지적할수 있다. 일감이름을 지적하면 그 일감은 전경일감으로 된다. 따라서 %1 은 fg %1 와 같은 의미를 가지며 일감 1 은 전경일감으로 된다. 또한 일감들을 실행시키려면 기호렬을 앞붙이로 하여 지적해 줄수 있다. 그 앞붙이들은 일의적이여야 한다. 따라서 이름이 기호렬 ex 로 시작되는 정지된 일감이 한개뿐이라면 %ex 는 정지된 일감 ex(1)을 재시동한다. %string 으로 string 을 포함하는 일감을 규정할수 있다.

csh 는 현재일감과 선행일감들의 특성들을 그대로 보존한다. 현재일감은 +로 표식되고 선행일감은 - 로 표식된다. 현재일감은 %%로도 표식될수 있다.

csh 는 프로쎄스상태가 변경되자마자 즉시 일감이 차단되여 더는 전진할수 없다는것을 입력재촉문을 인쇄하기전에 사용자에게 알려 준다. 이것은 사용자의 작업을 방해하지 않는다. 쉘변수 notify 를 설정하면 csh 는 배경일감들의 상태에서 변화가 있다는것을 사용자에게 알려 준다. 하나의 프로쎄스를 표식하는 notify 라고 부르는 csh 내장지령도 있으므로 임의의 상태변화를 즉시 알수 있다. 기정으로 notify 는 현재 프로쎄스를 표시한다. 배경일감을 시동시킨후 그것을 표시하려면 간단히 notify 를 입력하여야 한다.

일감들이 정지되였을 때 쉘에서 탈퇴하려고 하면 csh 는 정지된 일감이 있다는 경고통보문을 보낸다. 그 일감이 무엇인가를 보려면 지령 jobs 을 리용하여야 한다. 지령 jobs 을 주든가 또는 다시 탈퇴하려고 하면 csh 는 두번째 경고를 하지 않으며 정지된 일감들을 종결시킨다(exit(2)를 참고).

내장지령들

내장지령들은 새로운 프로쎄스를 파생함이 없이 쉘안에서 실행된다. 한 내장지령이 파이프선의 성분으로(마지막은 아니고) 출현하면 그것은 부분쉘에서 실행된다. 내장지령들은 다음과 같다.

alias
alias name
alias name wordlist

첫번째 형식은 모든 별명들을 인쇄하며 두번째 형식은 name 에 대한 별명을 인쇄한다. 세번째 형식은 규정된 wordlist 를 name 의 별명으로 대입한다. 지령과 파일이름치환은 wordlist 상에서 수행된다.

bg[%job…]

이 지령은 현재일감(규정되지 않은 일감) 또는 규정된 일감들을 배경

일감으로 한다. 그 일감들이 정지되여 있었다면 실행을 계속하게 한다.

break

이 지령은 가장 가까운 foreach 또는 while 의 뒤에서 실행을 계속하게 한다. 현재 행에 있는 나머지 지령들은 수행된다. 그러므로 한행에 여러개의 break 를 삽입하면 **다중준위중지**(Multi-Level Break)가 수행되게 된다.

breaksw

이 지령은 switch로부터 탈퇴하여 endsw의 뒤에서 실행을 계속하게 한다.

case label:

switch명령문에서의 한가지 표식

cd

cd directory_name

chdir

chdir directory_name

이 지령은 쉘의 현재작업등록부를 directory_name 으로 변경시킨다. directory_name 이 규정되지 않으면 기정으로 홈등록부가 리용된다. directory_name 이 현재작업등록부의 보조등록부로 나타나지 않으면(그리고 /, ./, ../ 으로 시작되지 않으면) 변수 cdpath 의 매 성분이 보조등록부 directory_name 을 가지는가를 검사한다. 모든것들이 실패하면 csh 는 directory_name 을 쉘변수로 취급한다. 그 값이 /으로 시작되면 그것이 등록부인가를 알아 보려고 시도한다.

continue

이 지령은 가장 가까이에 있는 while 또는 foreach 의 실행을 계속하게 한다. 현재행에 있는 나머지 지령들은 실행된다.

default:

switch 명령문에서 default 경우를 표식한다. 기정은 모든 다른 case 표식들의 뒤로 오는것이다.

dirs

이 지령은 등록부탄창을 인쇄한다. 탄창의 맨 꼭대기는 왼쪽에 있다. 탄창에 있는 첫 등록부는 현재등록부이다.

echo wordlist

echo ?n wordlist

규정된 단어들이 공백으로 분리되며 선택항목 ?n 이 규정되지 않으면 행바꾸기로 종결되며 쉘의 표준출력에 씌여 진다.

else

end

endif

endsw

foreach, if, switch, while 명령문들에 대한 해설을 참고

eval arguments…

sh(1)과 동일하게 작용한다. 이 지령은 지령치환 또는 변수치환의 결과로 발생되는 지령들을 실행시키는데 리용된다.

exec command

규정된 지령이 현재쉘대신에 실행된다.

exit

exit(expression)

csh 는 상태변수의 값을 가지고(첫번째 형식) 탈퇴하거나 규정된 식의 값을 가지고(두번째 형식) 탈퇴한다.

fg[%job…]

현재일감(규정되지 않은 일감) 또는 규정된 일감들을 전경에 가져 온다. 그것들이 정지되여 있으면 실행을 계속한다.

foreach name(wordlist)

…

end

변수 name 은 wordlist 의 매 원소들로 설정되고 foreach 와 end 사이에 있는 지령들의 렬이 실행된다. (foreach 와 end 는 서로 다른 행에 홀로 나타나야 한다.) 순환을 계속하기 위하여 내장지령 continue 를 리용할 수 있다. 내장지령 break 는 순환을 끝내기 위하여 리용될수 있다.
이 지령이 말단으로부터 읽혀 질 때 그 순환은 한번 읽혀 지며 순환안의 명령문들이 수행되기전에 ?로 재촉한다. 순환프로쎄스에 말단에서 입력하면서 오유를 범하면 삭제기호 또는 행제거기호를 리용하여 그것을 수정할수 있다.

glob wordlist

이 지령은 echo 와 류사하지만 \escape 들은 인식되지 않으며 단어들은 출력에서 null 기호들에 의하여 경계 지어 진다. 이 지령은 쉘을 리용하

여 wordlist 우에서 파일이름확장을 수행하는 프로그람에서 쓸모 있다.

goto word

word 는 표식형태의 기호렬을 주기 위하여 확장된 파일이름 또는 지령이다. 쉘은 word 로 표식된 행을 탐색한다. 실행은 그 행의 뒤에서 계속된다.

hashstat

이 지령은 내부적인 **하쉬표**(Hash Table)가 지령들에서 얼마나 효과적으로 리용되였는가를 가리키는 통계행을 인쇄한다.

history [-h] [-r] [n]

리력사건목록(History Event List)을 현시한다. n 이 주어 지면 가장 최근의 n 개 사건들이 인쇄된다. 선택항목 -r 는 가장 최근의것을 첫번째로 하는것이 아니라 가장 오랜것이 첫번째로 되도록 인쇄순서를 뒤집는다. 선택항목 -h 는 리력목록을 인쇄할 때 앞에 번호를 붙이지 않는다.

if(expression) command

expression 이 true 로 평가되면 인수들을 가지고 command 가 실행된다. command 상의 변수치환은 if 명령문이 아닌 다른 명령문에 대하여 변수치환이 진행될 때와 동일한 시각에 진행된다. command 는 단순지령이여야 하며 파이프선, 지령목록, 괄호안의 지령목록 또는 별명 붙은 지령은 될수 없다. 입출력방향바꾸기는 expression 이 false 일 때(이때는 command 가 실행되지 않는다.) 진행된다.

if (expression1) then

…

else if (expression2) then

…

else

…

end if

expression1 이 true 이면 첫 else 까지의 모든 지령들이 실행된다. 그렇지 않을 때 expression2 가 true 이면 첫번째 else 로부터 두번째 else 까지의 지령들이 실행된다. 임의의 개수의 else-if 쌍이 가능하지만 하나의 endif 만이 요구된다. else 부분은 생략가능하다(else 와 endif 는 입력행의 시작위치에 나타나야 한다. if 는 입력행 또는 else 뒤에서 홀로 나타나야 한다.).

jobs [-l]

작업중에 있는 일감들을 렬거한다. 선택항목 -l 는 보통의 정보외에 프로쎄스 ID 들을 렬거한다.

kill %job

kill - sig %job

kill pid

kill - sig pid…

kill - l

우의 지령들은 TERM(종결)신호 또는 규정된 신호를 규정된 일감 또는 프로쎄스들에 보낸다. 신호들은 /usr/include /signal.h 에 주어 진 번호 또는 이름에 의하여 주어 진다. 신호이름들은 지령 kill -l 에 의하여 렬거된다. 인수가 없이 리용된 지령 kill 은 현재일감에 신호를 보내지 않는다. 송신되는 신호가 TERM 또는 HUP 이면 CONT 신호로 송신한다.

limit [-h] [resource] [maximum_use]

이 지령은 규정된 resource 상에서 maximum_use 개수를 넘지 않도록 창조되는 프로쎄스들과 현재프로쎄스에 대한 사용을 제한한다. maximum_use 가 규정되지 않으면 현재한계가 현시된다. resource 가 규정되지 않으면 모든 한계가 주어 진다. 기발 -h 가 규정되면 현재한계대신에 기정한계가 리용된다. 기정한계는 현재한계들의 값에 대한 한도를 정한다. 상급사용자만이 기정한계를 변경시킬수 있다. 그러나 사용자는 허용되는 범위안에서만 현재한계를 올리거나 내릴수 있다.

조종가능한 자원들은 현재 다음과 같은것들을 포함한다.

addresspace	프로쎄스를 위한 최대주소공간(바이트)
coredumpsize	창조된 가장 큰 **주기억쏟기**(Core Dump)의 크기
cputime	매 프로쎄스에 의하여 리용될 CPU 의 최대초수
datasize	프로그람본문의 끝을 넘어 허용되는 자료령역의 최대범위
descriptors	매 프로쎄스에 대하여 열린 파일들의 최대개수
filesize	창조될수 있는 가장 큰 파일
memoryuse	프로쎄스의 상주모임크기가 증가할수 있는 최대크기
stacksize	자동적으로 확장된 탄창령역의 최대크기

인수 maximum_use 는 척도인수(즉 k 또는 키로바이트(1024 바이트), m 또는 메가바이트, b 또는 블로크(ulimit 체계호출에 의하여 리용된 단위))를 붙인 류점수 또는 옹근수에 의하여 규정될수 있다. 자원이름들과 척도인수들에 대하여 일의적인 이름들이 앞붙이로 리용될수 있다(더 많은 정보를 알려면 ulimit 체계호출에 대한 문서를 참고).

login

이 지령은 **가입쉘**(Login Shell)을 /usr/bin/login 의 **구체례**(Instance)로 교체하여 그것을 종결시킨다. 이것은 sh(1)과의 호환성을 위하여 도입된 작업중지의 한가지 방법이다.

logout

이 지령은 가입쉘을 종결시키며 특히 ignoreeof 가 설정되여 있을 때 리용된다. 가입쉘이 아닌 작업프로쎄스에 대하여 동작하는 류사한 함수 bye 가 있다. 이 함수는 표준 BSD csh 의 부분이 아니며 앞으로의 방안들에서는 지원되지 않기때문에 리용하지 말것을 권고한다.

newgrp

이 지령은 호출자의 그룹을 변경시킨다(상세한것을 보려면 newgrp(1)을 참고). 새로운 쉘이 newgrp 에 의하여 실행되므로 현재쉘의 환경은 없어 진다.

nice

nice +number

nice command

nice +number command

첫번째 형식은 이 쉘에 대한 실행지령우선권을 4 로(기정) 설정한다. 두번째 형식은 그 우선권을 주어 진 수 number 로써 설정한다. 마지막 두개의 지령은 command 를 각각 우선권 4 와 number 로 실행시킨다. 알맞는 특권을 가진 사용자는 nice -number 를 리용하여 우선권을 높일수 있다.

nohup [command]

인수가 없는 nohup 는 쉘스크립트들에서 스크립트의 나머지 부분을 지체시키지 않게 하는데 리용된다. 인수가 있는 nohup 는 규정된 지령 command 를 지체시키지 않고 실행되게 한다. 이 지령은 &를 붙여 배경에서 실행시킨 모든 프로쎄스들에서 효과적으로 리용된다.

notify [job…]

이 지령은 현재일감(규정되지 않은 일감) 또는 규정된 일감들의 상태가 변경되였다는것을 쉘이 사용자에게 비동기적으로 통보하게 한다. 통보는 표준적으로 재촉되기전에 제공된다. 쉘변수 notify 가 설정되였으면 자동적으로 통보된다.

onintr [-] [label]

이 지령은 중단할 때 쉘의 작용을 조종한다. 인수가 없으면 onintr 는 중단할 때 쉘의 기정작용을 복귀한다. 기정작용은 쉘스크립트를 종결

시키거나 **말단입력준위**(Terminal Input Level)에로 돌아 가는것이다. -가 규정되면 모든 중단이 무시된다. 표식이 주어 지면 쉘은 중단이 일어날 때 goto 명령문을 실행하며 표식이 주어 지지 않으면 후손프로쎄스가 종결된다. 쉘이 배경에서 실행되고 있고 중단이 무시되면 onintr 는 효력을 가지지 않는다. 즉 쉘과 호출된 모든 지령들은 중단들을 무시한다.

popd [+n]

이 지령은 등록부탄창의 맨 웃원소를 꺼내여 그것을 새로운 제일 웃등록부로 한다. 인수가 있으면 탄창에 있는 n 번째 항목이 꺼내여 진다. 등록부탄창의 원소들은 맨 우에서 시작하여 0 부터 번호가 매겨 져 있다. popd 와 기능이 같은 rd 가 마련되였다. rd 는 표준 BSD csh 의 부분이 아니며 앞으로의 방안들에서는 지원되지 않기때문에 리용하지 말것을 권고한다.

pushd [name] [+n]

이 지령은 인수가 없으면 등록부탄창의 두개의 맨 꼭대기원소들을 서로 교체한다. 인수 name 이 주어 지면 pushd 는 그것을 새로운 작업등록부로 변경(cd 를 리용하여)하고 낡은 작업등록부를 등록부탄창에 넣는다. 수값인수를 주면 pushd 는 등록부탄창의 n 번째 인수를 맨 꼭대기원소로 되도록 회전시키고 그 등록부를 변경시킨다. 등록부탄창의 원소들은 맨 꼭대기에서 시작하여 0 부터 번호가 매겨 져 있다. pushd 와 기능이 같은 gd 가 마련되였다. gd 는 표준 BSD csh 의 부분이 아니고 앞으로의 방안들에서 지원되지 않기때문에 리용하지 말것을 권고한다.

rehash

이 지령은 변수 path 에 있는 등록부의 내용들에 대한 내부적인 하쉬표를 다시 계산하게 한다. 이것은 사용자가 자기의 등록부에 지령들을 첨가하거나 체계프로그람작성자가 체계등록부의 내용들을 변경시킬 때 필요된다.

repeat count command

이 지령은 규정된 지령 command 를 count 로 지적한 회수만큼 실행시킨다. I/O 방향바꾸기는 count 가 0 일 때 정확히 한번만 진행된다.

set

set name

set name=word

set name[index]=word

set name=(wordlist)

첫번째 형식은 모든 쉘변수들의 값을 보여 준다.

값이 한개 단어가 아닌 변수들은 괄호안에 단어목록으로 인쇄된다. 두번째 형식은 name 을 null 기호렬로 설정한다. 세번째 형식은 name 을 한개 단어 word 로 설정한다. 네번째 형식은 name 의 index 번째의 성분을 word 에 설정한다. 이 성분은 항상 존재하여야 한다. 마지막형식은 name 을 wordlist 에 있는 단어들의 목록으로 설정한다. 모든 경우에 그 값은 확장된 지령 및 파일이름이다.

하나의 set 지령에서 여러개의 값을 설정하기 위하여 인수들을 반복하여 줄수 있다. 그러나 변수확장은 설정되기전에 모든 인수들에 대하여 일어 난다는것을 주의하여야 한다.

setenv name value

이 지령은 환경변수 name 의 값을 value 가 되도록 설정한다. 가장 흔히 리용되는 변수들인 USER, TERM, PATH 는 자동적으로 csh 변수 USER, TERM, PATH 에 전달되거나 그것으로부터 전달 받는다.

shift [variable]

인수가 주어 지지 않으면 argv 의 원소들은 왼쪽으로 밀리운다. argv 가 설정되지 않았거나 두개보다 적은 기호렬들을 가지면 오유가 발생된다. variable 이 규정되면 shift 는 규정된 변수에 관하여 동일한 기능을 수행한다.

source [-h] name

csh 는 name 으로부터 지령을 읽는다. source 지령들은 겹칠수 있으나 너무 많이 겹쳐 지면 쉘은 파일서술자들을 벗어 난다. 어느 한 source 지령에 오유가 있으면 모든 겹쳐 진 source 지령들을 종결시킨다. 표준적으로 source 지령을 통한 입력은 리력목록에 배치되지 않는다. 선택항목 -h 가 리용되면 지령들을 실행하지 않고 리력목록에 배치한다.

stop[%job…]

이 지령은 인수가 없으면 현재 일감을 정지시키고 인수가 있으면 배경일감을 정지시킨다.

suspend

이 지령은 csh 가 정지신호를 받은것처럼 정지되게 한다. csh 는 표준적으로 정지신호를 무시하기때문에 이 지령은 쉘을 정지시키기 위한 한가지 방법일뿐이다. 가입쉘로부터 이 지령을 호출하면 오유가 발생한다.

```
switch(string)
case   str1:
…
breaksw
…
default:
…
breaksw
endsw
```

string 은 지령 및 파일이름일수 있다. 매 case 표식(str1)이 string 에 맞으면 그 case 표식뒤의 지령들이 수행된다. default 표식앞에 있는 표식들중에서 맞는것이 없으면 default 표식뒤의 지령들이 수행된다. 매 case 표식과 default 표식은 행의 앞부분에 나타나야 한다.

지령 breaksw 는 endsw 뒤에로 이행하게 한다. breaksw 를 기입하지 않으면 C 에서처럼 그 아래의 case 표식들과 default 표식들뒤의 지령들을 실행한다. 맞는 case 표식이 없고 default 표식이 없으면 실행은 endsw 뒤에서 계속된다.

time [command]

command 가 규정되지 않으면 쉘과 그 후손들에 의하여 리용되는 시간들이 인쇄된다. 만일 command 가 규정되면 그 단순지령의 시간이 측정되고 변수 time 에 서술된대로 시간개괄이 인쇄된다. 필요하면 지령이 완결되였을 때 시간통계를 인쇄하기 위한 쉘이 따로 창조된다.

umask [value]

value 가 규정되지 않았을 때에는 현재파일창조마스크가 현시되며 value 가 규정되였을 때에는 규정된 value 로 설정한다. 마스크는 8 진수로 주어 진다. 공통적인 값은 002 이다. 이것은 소유자와 그룹에게는 모든 허락들을 주며 기타 사람들에게는 읽기와 실행허락을 준다.

공통적인 값은 또한 022 이다. 이것은 소유자에게는 모든 허락을 주고 그룹과 기타 사람들에게는 읽기와 실행허락만을 준다.

unalias pattern

이 지령은 규정된 pattern 에 맞는 별명들을 버린다. 실례로 unalias *은 모든 별명들을 제거한다. 패턴에 맞는 별명이 존재하지 않으면 오유가 발생하지 않는다.

unhash

이 지령은 실행된 프로그람의 위치선정을 빨리 하게 하는 내부적인 하쉬표를 리용할수 없게 한다.

unset pattern

이 지령은 규정된 패턴 pattern 에 맞는 별명들을 제거한다. 실례로 unset *은 모든 명령들을 제거한다. 패턴에 맞는 명령이 존재하지 않으면 오유가 발생하지 않는다.

unsetenv pattern

이 지령은 규정된 패턴 pattern 에 맞는 모든 변수들을 환경으로부터 제거한다(setenv 지령과 printenv(1)을 참고).

wait

이 지령은 모든 배경일감이 종결되기를 기다린다. 쉘이 대화형이면 중단을 일으켜 완결되지 않은 모든 일감들의 이름들과 번호들을 인쇄하는 동안 기다리지 않게 할수 있다.

while（expression）

…

end

규정된 식 expression 이 령이 아닌 동안 while 과 end 사이에 있는 지령들이 수행된다. 순환을 종결시키거나 계속하기 위하여서는 break 와 continue 를 리용할수 있다. while 과 end 는 입력행에서 홀로 나타나야 한다.

%job

이 지령은 규정된 일감을 전경에서 실행시킨다.

%job &

이 지령은 규정된 일감이 배경에서 계속 실행되게 한다.

@

@ name=expression

@ name [index] = expression

첫번째 형식은 모든 쉘변수들의 값을 인쇄한다. 두번째 형식은 규정된 이름을 식의 값으로 설정한다. 식이 <, >, & 또는 | 을 포함하면 적어도 식의 그 부분은 괄호안에 놓여야 한다. 세번째 형식은 식의 값을 이름의 index 번째 인수에 대입한다. 이름과 index 번째 성분은 항상 존재하여야 한다.

연산자 *=, += 등은 C 에서처럼 리용될수 있다. 대입연산자와 name 사이에 공백이 놓일수 있다.

특수한 뒤붙이연산자 ++ 와 -- 는 각각 name 의 값을 증가시키거나 감소시킨다.

비내장지령의 실행

실행되여야 할 지령이 내장지령이 아니면 csh 는 그 지령을 exec(2)을 통하여 실행시킨다. 지령이 /으로 시작하지 않는 경우 변수경로에 있는 매 단어는 쉘이 지령을 찾으려고 하는 장소인 등록부를 이름 짓는다. -c 와 -t 가 주어 지지 않으면 쉘은 그 등록부에 있는 이름들을 내부표에 넣으며 exec 는 그 지령이 들어 있는 등록부에서만 실행된다. 이것은 많은 등록부들이 탐색경로에 존재할 때 지령의 위치설정을 현저히 빠르게 한다. 이 동작이 진행되지 않거나, -c 또는 -t 가 주어 져 있거나, 경로의 등록부성분이 /로 시작되지 않으면 쉘은 등록부이름과 주어 진 지령이름을 련결하여 실행해야 할 파일의 경로이름을 완성한다.

괄호안에 있는 지령들은 항상 부분쉘에서 실행된다. 따라서

(cd ; pwd)

는 홈등록부를 인쇄하고 완결된후에 현재등록부로 돌아 온다. 반대로

cd ; pwd

는 완결된 후에 홈등록부에 머무른다.

지령들이 괄호안에 들어 있으면 chdir 는 보통 현재쉘에 영향을 주지 않는다.

파일이 실행허락을 가지지만 실행가능한 2 진파일이 아니면 그 파일은 분리된 프로쎄스로 실행되는 해석기를 위한 자료파일인 스크립트파일일것이라고 가정한다. csh 는 우선 스크립트파일을 적재하고 실행시키려고 시도한다(exec(2)를 참고).

그 스크립트파일의 첫 두개기호가 #!이면 exec(2) 은 그 뒤에 해석기경로이름이 있을것을 요구하며 규정된 해석기를 전체 스크립트파일을 읽기 위한 분리된 프로쎄스로서 실행시키려고 한다.

해석기가 #!로 이름 지어 져 있지 않으며 그 쉘에 대한 별명이 존재하면 그 별명의 단어들은 인수목록의 앞에 삽입되여 쉘지령을 형성한다. 별명의 첫 단어는 리용될 지령의 경로이름일수 있다. 이것은 단어들을 변경시키지 않고 인수목록에 삽입하는 별명치환의 특수경우이다.

해석기가 #!로 이름 지어 져 있지 않으며 쉘별명이 없지만 그 파일의 첫 기호가 #이면 $shell 변수에 의하여 이름 지어 진 해석기가 실행된다. (이것은 사용자가 $shell 을 재설정하지 않는 한 표준적으로 /usr/bin/csh 이다.) $shell 이 설정되여 있지 않으면 /usr/bin/csh 가 실행된다.

해석기가 #!로 이름 지어 져 있지 않으며 쉘별명이 없고 그 파일의 첫 기호가 #이 아니면 그 스크립트파일을 해석하기 위하여 /usr/bin/csh 가 실행된다.

리력치환

리력치환은 지령을 반복하고 선행지령들의 단어들을 새 지령의 일부분으로 리용하며 선행지령의 인수들을 현지령에서 반복하고 맞춤법, 또는 입력오유를 고치는데 리용될수 있다.

리력치환은 감탄부호(!)로 시작된다. 치환은 입구흐름의 그 어디에서나 시작할수 있지만 겹칠수는 없다. 감탄부호앞에 거꿀빗선(\)이 놓이면 그것의 특수한 의미가 없어 진다. 편의상 감탄부호뒤에 공백, tab, 행바꾸기, = 또는 (가 놓이면 감탄부호는 변경되지 않고 문장론해석기에 전달된다. 지령치환을 포함하는 입구행들은 실행되기전에 말단에 출력된다.

말단으로부터 입력된 한개 또는 그이상의 단어들로 이루어 진 지령들은 리력목록에 보관된다. 리력치환은 그 보관된 지령들로부터의 단어들의 렬을 입구흐름에 다시 가져 온다. 보관되는 선행지령들의 개수는 변수 history 에 의하여 조종된다. 선행지령은 그 변수의 값을 고려함이없이 항상 보관된다. 지령은 1 부터 순차적으로 번호화된다.

선행사건들은 사건번호(사건 10 에 대하여 !10 과 같이), 상대적인 사건위치(두번째 선행한 사건에 대하여 !-2 와 같이), 전반적인 또는 부분적인 지령이름(초기기호가 d 인 지령을 리용하는 마지막사건에 대하여 !d 와 같이), 기호렬식(기호 mic 들을 포함한 사건을 가리키는 !?mic?와 같이)으로 지적될수 있다.

이 형식들은 다른 변경이 없이 규정된 사건들의 단어들을 간단히 다시 끌어 들인다. 특수한 경우로서 !!는 반복이다. 즉 이것은 선행지령을 지적한다.

지령으로부터 단어를 선택하려면 두점(:)과 사건규정후에 요구되는 단어지명기호를 리용한다. 한개 입구행의 단어들은 0 부터 번호가 매겨 져 있다. 기본적인 단어지령기호들은 다음과 같다.

0	첫 단어 (즉 지령이름 차례)
n	n 번째 단어
^	첫번째 인수(이것은 1 과 동등하다.)
$	마지막단어
a-b	a 부터 b 까지의 단어들의 범위
*	두번째 단어로부터 마지막단어까지의 범위
%	바로 직전의 맞는 단어를 치환하기 위한 탐색렬과 함께 리용한다.

지령규정을 단어지명기호와 분리시키는 두점은 시작선택기호가 ^, $, *, -, 또는 %로 시작되는 경우에 생략될수 있다.

단어지명기호뒤에는 조작기호렬이 놓일수 있다. 다음과 같은 조작기호들이 정의되여 있다.

h 경로이름의 첫 성분만을 리용한다. 그 뒤의 성분들은 모두 제거

된다.

r	꼬리에 붙은 뒤붙이(.xxx)를 제거하고 뿌리파일이름을 리용한다.
e	뿌리이름을 제거하고 파일이름의 꼬리에 붙은 뒤붙이를 리용한다.
s/l/r	지적된 지령에서 값 l에 대하여 r의 값을 치환한다.
t	경로이름의 마지막파일이름만을 리용한다. 앞부분의 경로이름성분들을 제거한다.
&	선행치환을 반복한다.
p	새 지령을 인쇄만 하고 실행시키지 않는다.
q	치환된 단어들을 인용한다. 앞으로의 치환을 방지한다.
x	q와 같으나 공백, tab, 행바꾸기들에서 중지된다.
g	대역적지령을 다른 조작기호의 앞붙이로 리용하여 지적된 변경이 대역적으로 진행되도록 한다. 그 지령에 있는 모든 단어들이 변경되며 웃반점(') 또는 웃두점(")안에 있는 기호렬은 하나의 단어로 취급된다.

g가 앞에 붙지 않으면 조작은 조작가능한 첫 단어에만 적용된다. 치환이 완성될수 없으면(즉 10개의 지령들만을 포함하는 리력완충기상에서 치환 !11을 시도하면) 오유가 발생한다.

치환의 왼쪽면은 기호렬이다. 즉 HP-UX 편집기들의 경우에 **정규식**(regular expression)이 아니다. 빗선(/)대신에 임의의 기호들이 제한기호로 리용될수 있다. 제한기호가 l 또는 r 기호렬에 리용되면 그것을 인용하는데 거꿀빗선(\)을 리용한다. 오른쪽에 있는 기호 &는 왼쪽에 있는 본문에 의하여 교체된다. \로 &을 인용한다. 치환에서 꼬리에 붙인 제한기호는 행바꾸기가 곧 있을 때 생략될수 있다.

사건을 규정함이 없이 리력을 참조할수 있다. 이 경우에 선행한 리력참조가 동일한 행우에서 진행되지 않는한 선행지령이 참조된다. 따라서

```
!?foo?^   !$
```

는 ?foo?에 맞는 지령으로부터 첫번째 인수와 두번째 인수를 준다.

입력행에서 공백이 아닌 첫 기호가 ^이면 이것은 리력참조가 생략된다는것을 의미한다. 이것은 선행한 행의 본문상에서 치환을 쉽게 하게 하는 1:s^과 동등하다.

끝으로 리력치환을 뒤따르는 기호들과 분리시킬 필요가 있으면 그것을 대괄호 { }안에 넣을수 있다. 따라서

```
ls -ld ~paul
```

의 뒤에서 !{l}a를 실행시켜

```
ls -ld ~paula
```

가 되게 할수 있다. 한편 !la는 la로 시작되는 지령처럼 보인다.

제１２장. 쉘프로그람작성에 대한 개괄

쉘프로그람작성

쉘에는 눈에 보이는것보다 보이지 않는것이 더 많다. 쉘은 많은 사람들에게 습관되여 있는 지령행해석기보다 더 많은 기능을 수행할수 있다. UNIX 쉘들은 위력한 해석형 프로그람작성언어도 제공한다.

이 장에서는 ksh 쉘프로그람작성방법에 대하여 취급한다. ksh 를 선택한 리유는 대부분의 ksh 프로그람작성기술들이 Bourne 쉘에서도 동작하기때문이다. ksh 프로그람작성방법을 서술한데 이어 이 장의 마지막부분에서 csh 프로그람작성방법에 대하여 취급한다. csh 는 ksh 와는 다른 프로그람작성기술을 리용한다.

여기서는 중요한 ksh 프로그람작성기술들에 대하여 취급한다. 이 장에서 취급되는 내용들중에서 가장 중요한것은 rm 으로 파일들을 체계로부터 영원히 제거하는것이 아니라 파일들을 제거하면서 그것들을 등록부에 배치하는 쉘프로그람이다. 이 프로그람은 이 장에서 취급되는 모든 쉘프로그람작성기술들을 리용한다. trash 라고 부르는 이 쉘프로그람은 약간 변경되면 Bourne 쉘에서도 수행될수 있다.

쉘은 임의의 UNIX 체계들에서 가장 중요한 특성들중의 하나이다. 만일 과제를 완료하기 위한 정확한 지령을 찾을수 없는 경우에는 쉘스크립트를 리용하여 쉽게 그 과제를 완료할수 있다.

쉘프로그람작성법을 배우기 위한 가장 좋은 방법은 실례를 통하여 배우는것이다. 이 장에서 취급되는 일부 실례들에서는 현재 론의되는 문제에 대해서만 설명하였다. 그러나 대부분의 실례들에서는 쉽게 확장하여 다른 환경과 결합할수 있게 하는 쓸모 있는 수단들과 그 부분들에 대해서도 보여 주고 있다. 그 실례들에는 쉽게 리해할수 있게 하는 예고문들과 출력통보문들이 있다. 대부분의 실례들은 어떤 기능을 수행하려고 하는 경우에 무엇이 필요한가를 보여 주고 있지만 오유검사는 하지 않는다. 프로 그람이 수행되는데는 몇분밖에 걸리지 않으며 매 환경과 오유조건을 처리하는데는 많은 시간이 걸린다.

이 장에 있는 실례들중의 대부분은 Solaris 체계에서 실행할수 있게 되여 있다. 그러나 쉘들은 한 체계로부터 다른 체계에로 거의 동일하게 절환되기때문에 이 장에 있는 프로그람들을 다른 체계에서도 실행시킬수 있다.

쉘프로그람을 만들기 위한 단계

모든 쉘프로그람들에서 일관성을 보장하기 위하여서는 쉘프로그람을 작성할 때 몇개의 초기화단계를 거쳐야 한다. 아래에 쉘프로그람에서 리용되는 몇가지 중요한 개념들을 설명한다.

1. 쉘프로그람들과 출력의 이름

쉘프로그람과 그 출력에 의미있는 이름을 붙여 주어야 한다. 만일 쉘프로그람들을 script1, script2, … 등으로 부르면 이것은 작성자나 다른 사용자들을 위해서도 그 어떤 의미를 가지지 않는다. 만일 쉘프로그람이 특수한 형의 파일들을 찾는다면 그 프로그람을 filefind 또는 어떤 다른 의미있는 이름으로 붙여 줄수 있다. 쉘프로그람의 출력에 대하여서도 마찬가지이다. 쉘프로그람의 출력을 output1 이라고 부르지 말고 어떤 의미있는 이름 즉 filefind.out 라고 붙여 주는것이 좋다. 이미 존재하는 쉘프로그람과 그 출력에 이름을 붙여 주는것은 피하여야 한다. 실례로 지령 read, test 와같은 지령들을 리용하면서 프로그람과 그 출력에 이름을 붙여 주는 경우에는 혼돈과 충돌을 일으킬수 있다. 이 장에서 취급된 첫 쉘프로그람은 오늘의 날자를 보여 준다. 그 프로그람을 date 라고 이름지었다면 이것은 쉘프로그람 date 에 대립되는것으로써 현재체계의 date 지령을 실행시키게 된다. 왜냐하면 대부분의 경우에 체계의 date 지령이 쉘프로그람 date 보다 먼저 발견되기때문이다.

2. 쉘프로그람들을 위한 저장고

만일 여러개의 쉘프로그람들을 작성하려는 경우 그것들을 어느한 등록부에 넣을수 있다. 작성자의 홈등록부에 bin 또는 shellprogs 라고 부르는 등록부가 있다면 쉘프로그람이 배치되여 있는 그 등록부이름을 포함하도록 경로를 갱신할수 있다.

3. 쉘이 프로그람을 실행시키도록 규정

프로그람의 제일 첫 행은 프로그람을 실행시키는데 리용되여야 할 쉘을 규정한다. 쉘의 경로의 맨 앞에는 기호 #!가 붙는다. 이 기호는 그 뒤에 놓이는것이 쉘의 경로라는것을 가리킨다. 이 장의 쉘프로그람들은 ksh 경로를 리용한다.

4. 쉘프로그람들의 형식화

쉘프로그람들을 읽기 쉽게 하자면 적당히 형식화하는것이 중요하다. 프로그람의 령역을 약간 들여 다 기입하여 지령들이 한 그룹의 부분이라는것을 알수 있게 하여야 한다. 3~4 개 칸정도로 들여 다 쓰는것이 좋다. 또한 자동들여쓰기를 vi 에 설정하여 다음행이 선행한 행과 동일한 위치에서 시작하도록 할수 있다(이 기술이 어떻게 동작하는가를 보려면 vi 에서 :set ai 를 실행시켜 보시오.). 긴 행을 끊으려면 행의 뒤에 거꿀빗선(\)을 치고 다음 행에서 계속하면 된다.

5. 설명문

쉘프로그람에 상세한 설명문을 포함시켜 프로그람의 어느한 부분이 무엇을 수행하는가를 서술해 줄수 있다. 이것은 작성자가 프로그람을 몇달 또는 몇년후에 볼 때 또는 다른 사람들이 그 프로그람들을 볼 때 쉽게 리해하도록 한다. 설명문에 포함시킬수 있는 량에는 한계가 없다. 설명문은 기호(#)로 시작한다.

6. 쉘프로그람을 실행가능하게 하기

앞장에서 다양한 쉘들을 도입하면서 취급한 chmod 지령은 프로그람을 실행가능하게 하는데 리용된다.

이렇게 하면 쉘프로그람의 창조와 리용은 아주 편리해 진다. 다음으로 쉘프로그람의 형태에 대하여 취급한다.

쉘프로그람은 순차적으로 수행시키려는 지령들을 포함하고 있는 파일이다. 그 파일이 실행허락으로 설정되여 있으면 스크립트의 이름을 입력하자마자 곧 실행된다.

쉘프로그람은 다음과 같은 두가지 기본형식을 가진다.

1. 단순지령파일

   지령행 또는 반복적으로 리용되는 지령들의 모임이 있으면 그것들을 모두 수행시키는데 한개 단순지령을 리용할수 있다.

2. 구조화된 프로그람

   쉘은 지령들의 렬을 《묶어 놓는》 가능성보다 더 많은 기능을 제공한다. 쉘은 다른 고급프로그람작성언어들에 있는 특성들을 많이 가지고 있다. 즉

   · 자료기억을 위한 변수들
   · 결심채택조종들 (if 지령과 case 지령)
   · 순환가능성(for 순환과 while 순환)
   · 모듈화를 위한 함수호출

이 두가지 기본형식이 있기때문에 사용자들은 단순한 지령교체로부터 복잡한 자료조작과 체계관리도구들에 이르기까지 모든것을 만들수 있다.

ksh 프로그람작성

쉘프로그람을 보관하기 위한 장소로 리용하기 위하여 홈등록부에 shellprogs 라고 부르는 등록부를 창조한다. 절대경로를 규정함이 없이 프로그람들을 실행시키려면 .profile 에 있는 경로에 다음의 행을 포함시키면 된다.

```
export   PATH = ${PATH} : ~ shellprogs
```

이것은 경로에 /home/martyp/shellprogs 를 첨가한다. 언제나 그러하듯이 두점(:)은 분리기호이다. 경로가 쉘프로그람이 보관되여 있는 등록부를 포함하는가를 확인하려면 아래의 지령을 실행시키면 된다.

```
martyp   $   echo   $PATH
/usr/bin:/usr/ucb:/etc:.:/home/martyp/shellprogs
martyp   $
```

이제 shellprogs 등록부에 가서 today 라고 부르는 단순지령파일을 다음과 같이 입력하자.

```
#!/bin/ksh
# 이것은 오늘의 날자를 현시하는 단순지령파일이다.
echo " Today's date is "
date +%x
```

프로그람의 매 행을 보기전에 그 프로그람을 실행시켜 결과를 보자.

```
martyp $ today
ksh: today : cannot execute
martyp $ ls -al
total 6
drwxrwxr-x2  martyp   staff   512  May  21  09:53   .
drwxrwx--- 4 martyp   staff   512  May  21  09:25   . .
-rw-rw-r--  1 martyp  staff   100  May  21  09:54   today
martyp $ chmod +x today
martyp $ ls -al
total 6
drwxrwxr-x2  martyp   staff   512  May  21  09:53   .
drwxrwx--- 4 martyp   staff   512  May  21  09:25   . .
-rwxrwxr-x 1 martyp   staff   100  May  21  09:54   today
martyp $ today
Today's date is
05/21/99
martyp $
```

이 결과에서 확인할수 있는바와 같이 today 가 실행허락이 없이 창조된 파일이기때문에 그것을 실행시킬수 없으나 지령 chmod +x 로 그 파일에 실행허락을 추가한 다음에는 today 를 실행시킬수 있고 그 결과를 볼수 있다는것을 알수 있다.

앞장들에 있는 umask 에 대한 론의는 새 파일들에 대한 기정값들을 서술한다. 새 파일들은 거의나 실행허락이 없이 창조되므로 파일에 실행허락이 포함되도록 변경시켜야 한다.

이제는 우의 쉘프로그람을 따라 가면서 매 행들을 해석해 보기로 한다.

```
#!/bin/ksh
```

첫번째 행은 ksh 가 리용되게 된다는것을 규정한다. 만일 Bash, C 쉘 또는 임의의 다른 쉘을 리용한다면 그것의 위치를 이러한 방식으로 규정해 주어야 한다. 일부 체계들은 그 우에서 실행되는 여러가지 쉘방안들을 가지고 있다. 왜냐하면 쉘이 마지막으로 출현한 때로부터 갱신되였으며 일부 사용자들은 그 쉘의 낡은 방안을 보존하려고 하였기때문

이다. 이러한 리유로 하여 규정된 절대경로가 실지 리용하려고 하는 쉘의 경로라는것을 담보해 주어야 한다. 파일의 제일 첫 두개 기호는 #!이여야 한다.

일반적으로 쉘프로그람을 실행시킬 때 체계는 대화적지령행을 위해 리용하고 있는것과 동일한 쉘을 리용하여 지령들을 실행시키려고 한다. 만일 이런 행을 포함시키지 않는 경우에 ksh 가 아닌 다른 쉘을 리용하고 있는 사람들은 어느한 ksh 프로그람을 실행시키려고 할 때 기대하지 않았던 결과를 얻게 된다.

좋기는 작성되는 매 쉘프로그람의 첫 행으로 #!shellname 을 포함시키는것이다.

이제 프로그람의 다음 행을 보자.

```
# 이것은 오늘의 날자를 현시하는 단순지령파일이다.
```

이것은 설명문이다. 한 지령행에서 #의 뒤에 있는 모든것들은 설명문으로 간주된다.(첫 행에 있는 #!은 제외된다.)

다음 지령은

```
echo "Today's date is"
```

이다.

echo 지령은 쉘프로그람에서 재촉문과 통보문을 발생시킨다. 출력을 형식화하기 위하여 echo 에서 리용할수 있는 선택항목들을 보려면 echo 에 대한 안내서를 참고하여야 한다. 현시되여야 할 기호렬을 일반적으로 웃두점안에 넣는다. 이 경우에 웃반점이 기호렬의 한 부분이라는것을 쉘이 알도록 하기 위하여 인용부호로서 웃두점을 리용한다.

프로그람의 마지막지령은

```
date +%x
```

이다. 이것은 date 지령을 수행시킨다. date 지령에는 많은 선택항목들이 있는데 그중 일부는 직관성을 보장하지 못한다. 우의 경우에 가장 단순한 형식으로서 오늘의 날자를 얻기 위한 선택항목만을 리용하였다.

이제 쓸모 있는 지령파일의 실례를 하나 더 취급하자. 이 프로그람은 현작업등록부를 알려 주고 그 등록부에 있는 파일들을 보여 준다. 아래에 쉘프로그람 myll 을 보여 준다.

```
#! /bin/ksh
# This is a simple shell program that displays the current
# directory name before a long file listing (ll) of that
# directory.
# The script name is myll
echo "Long listing of directory:"
pwd
echo
ll –1
```

이 프로그람은 ll 을 리용한다. 사용자는 ls -al 을 줄 필요가 있다. 아래에 myll 이 실행된 결과를 보여 준다.

```
martyp $ myll
Long listing of directory:
/home/martyp/shellprogs

total   4
-rwxrwxr-x 1  martyp        staff     220  May  24  05:28  myll
-rwxrwxr-x 1  martyp        staff     100  May  21  09:54  today
martyp $
```

현재작업등록부의 이름은 /home/martyp/shellprogs 이다. 이 등록부의 내용들은 이 장에서 지금까지 취급한 두 프로그람들을 보여 준다.

보다 더 복잡한 쉘프로그람을 작성하기전에 쉘에 있는 프로그람작성특성들에 대하여 더 배울 필요가 있다. 쉘변수들로부터 시작하자.

쉘변수

쉘변수는 프로그람작성언어들의 변수와 류사하다. 변수는 단순히 기억위치에 주는 이름이다. 그러나 대부분의 언어들과는 달리 변수들을 소개하거나 초기화하지 않고 즉시 리용할수 있다.

쉘변수들은 문자(대문자 또는 소문자)로 시작되는 임의의 이름일수 있다. 특수한 쉘기호들(파일이름발생기호들)과의 혼돈을 피하기 위하여 이름을 단순하게 지우며 문자, 수자, 밑줄만을 리용하여야 한다.

쉘변수에 값을 대입하려면 간단하게 다음과 같이 기입하면 된다.

```
name=value
```

기호 =의 앞과 뒤에 공백이 있으면 안된다.

지령행으로부터 쉘변수들을 설정하는 몇가지 실례들을 고찰하자.

$ myname=ralph

$ HerName=mary

이 실례들은 정확히 동작한다. 그러나

$ his name=norton

his: not found

는 "his"뒤에 공백이 있기때문에 동작하지 않는다. 쉘은 "his"를 지령으로 가정하고 그것을 실행시키려고 한다. 그 행의 나머지 부분은 무시된다.

$ **one+one=two**
one+one=two: not found

이 실례에서는 이름에 무효한 기호(+)를 포함하고 있다. 변수는 문자로 시작되여야 한다. 변수에 정확한 의미를 주지만 문자로 시작되지 않은 이름을 주는것이 흔히 나타나는 오유이다.

$ **3word=hi**
3word=hi: not found

이 변수에 값을 대입하려고 할 때 "3"은 "not found"를 발생시킨다.

변수에 값을 대입하는 방법을 취급하였으므로 이제는 변수들을 리용하는 방법에 대하여 취급한다. $는 변수의 값을 얻는데 리용된다. 쉘은 지령행에서 $를 만날 때마다 그 뒤에 놓이는 기호들을 변수이름으로 가정한다. 쉘은 $variable 을 그 값으로 교체한다. 지령행에서 변수들을 리용하는 몇가지 실례를 고찰한다.

$ **myname=ralph**
$ **echo myname**
myname
$ **echo $myname**
ralph
$ **echo $abc123**

첫 echo 지령에는 $가 없으므로 쉘은 myname 을 무시하며 echo 는 myname 을 출력해야 할 인수로 본다. 그러나 두번째 echo 지령에는 $가 있으므로 myname 의 값을 지령행에 출력한다. echo 는 ralph(myname 또는 $myname 이 아니라)를 인수로 여긴다. 마지막 echo 명령문에서 abc123 은 값을 가지지 않으며 따라서 쉘은 $abc123 을 아무것으로도 교체하지 않는다. 그러므로 echo 는 인수를 가지지 않으며 빈행을 출력한다.

변수들과 기호렬들을 련결하고 싶을 때가 있다. 쉘에서 이것은 아주 쉽게 할수 있다.

$ **myname=ralph**
$ **echo "My name is $myname"**
My name is ralph

지령행에서 변수이름이 쉽게 식별되지 않을 때에는 쉘이 혼란에 빠질수 있다.

$ **string=dobeedobee**
$ **echo "$stringdoo"**

여기서는 "dobeedobee"를 현시하려고 하였는데 쉘은 변수이름이 stringdoo 라고 생각하였다. 이 변수는 값을 가지지 않는다. 이러한 오유가 생기지 않게 하려면 변수이름을 대괄호안에 넣어 주위의 기호들과 분리시켜야 한다.

```
$ echo "${string}doo"
dobeedobeedoo
```

쉘프로그람에 있는 변수들을 같은 방법으로 설정할수 있다. 그러나 지령의 출력을 변수에 보관하여 그것을 후에 리용하도록 할수 있다. 사용자에게 질문을 제기하고 그들의 응답을 변수에 읽어 넣고 그것을 조사할수 있다.

지령치환

지령치환은 지령(stdout)으로부터의 출력을 쉘변수에 보관할수 있게 한다. 이것을 설명하기 위하여 "today"실례가 지령치환을 리용하여 어떻게 되는가를 보기로 하자.

```
#!/bin/ksh
d=`date +%x`
echo "Today's date is $d"
```

date 지령을 둘러 싼 거꿀웃반점(`)은 쉘이 date 를 실행하고 그 출력을 지령행에 배치하게 한다. 그러면 그 출력은 변수 d 에 대입된다. 갱신된 이 스크립트를 today1 이라고 이름 지어 놓고 그것을 실행시킨다.

```
$ today1
Today's date is 05/24/00
```

이 문제를 변수 d 를 리용하지 않고도 처리할수 있다. 다음 실례에서 보여 주는 today2 스크립트에서와 같이 date 지령을 출력기호렬안에 포함시킬수도 있다.

```
#!/bin/ksh
echo "Today's date is `date +%x`"
```

이 프로그람을 실행시키면 today1 의것과 꼭같은 출력을 얻는다.

```
$ today2
Today's date is 05/24/00
```

쉘변수들과 지령치환은 뒤의 실례들에서 널리 리용된다. 다음으로 사용자입력의 읽기를 취급하자.

사용자입력의 읽기

사용자로부터의 정보를 얻기 위한 가장 공통적인 방법은 재촉문을 주고 응답을 읽는 것이다. echo 지령은 재촉문을 현시하는데 가장 흔히 리용되는 지령이다. 그다음 read 지령은 사용자로부터의 입력행을 읽는데 리용된다(표준입력).

입력행에 있는 단어들은 한개 또는 여러개의 쉘변수들에 대입된다. 다음의 실례에서는 read 가 어떻게 리용될수 있는가를 보여 주고 있다.

```
#! /bin/ksh
# program: readtest
echo "Please enter your name: \c" # the \c leaves cursor on
                                  # this line.
read name # I have no $ because we are doing an assignment
          # of whatever the user enters into name.
echo "Hello, $name"
echo "Please enter your two favorite colors: \c"
read colorl color2  # first word entered goes into color1
                    # remainder of line goes into color2
echo "You entered $color2 and $color1"
```

이 프로그람을 실행시키면 다음과 같은 결과가 얻어 진다.

$ **readtest**
Please enter your name: **gerry**
Hello, gerry
Please enter your two favorite colors: **blue green**
You entered green and blue
$

read 지령이 입력된 두개의 color 들을 어떻게 두개의 color 변수들에 각각 대입하였는가를 주목하여야 한다. 만일 사용자가 read 지령이 기대하였던것보다 적은 단어들을 입력하였다면 나머지 변수들은 null 로 설정된다. 만일 사용자가 너무 많은 단어들을 입력하면 남은 단어들은 모두 마지막변수에 대입된다. 다음의 실례에서는 두개이상의 color 를 입력할 때 어떤 현상이 일어 나는가를 보여 주고 있다.

$ **readtest**
Please enter your name: **gerry**
Hello, gerry
Please enter your two favorite colors: **chartreuse orchid**
blue You entered orchid blue and chartreuse
$

프로그람은 입력된 마지막두개의 color 를 선택하여 그것들을 color2 에 대입하였다. 이 리유로부터 사용자는 read 지령으로 입력하는것에 대하여서와 그 정보가 변수들에 어떻게 대입되는가에 대하여 주의를 돌려야 한다.

read 와 함께 리용될수 있는 REPLY 라고 부르는 내장변수가 있다. REPLY 를 쓰면 입력행이 그 변수에 대입된다. 다음의 실례는 readtest1 과 그 결과를 보여 준다.

```
martyp $ cat readtest1

#!/bin/ksh
# program: readtest
echo "Please enter your name: \c" # the \c leaves cursor on
                                  # this line.

read name # I have no $ because we are doing an assignment
          # of whatever the user enters into name.
echo "Hello, $name"
echo "Please enter your two favorite colors: \c"
read colorl color2  # first word entered goes into color1
                    # remainder of line goes into color2

echo "You entered $color2 and $color1"
echo "Where are you from?"
read                # read response into $REPLY
echo "I'm sure $REPLY is great"

martyp $ readtest1
Please enter your name: MARTY
Hello, MARTY
Please enter your two favorite colors: RED BLUE
You entered BLUE and RED
Where are you from?
MANY DIFFERENT PLACES
I'm sure MANY DIFFERENT PLACES is great
martyp $
```

이 실례에서 보는바와 같이 사용자의 응답 "MANY DIFFERENT PLACES"는 REPLY 에 읽혀 졌다.

이 장에서는 echo 를 리용하였지만 화면에 행들을 현시하는데는 print 도 리용될수 있다.

쉘프로그람에 주는 인수

쉘프로그람들은 **정규지령**(Regular Command)들처럼 지령행인수들을 가질수 있다. 쉘프로그람을 작성할 때 리용되는 지령행인수들은 특수한 변수들의 모임에 보관된다. 그것들을 **위치적파라메터**(Positional Parameter)라고 부른다. 지령행의 첫 10개 단어들은 쉘프로그람에서 특수한 변수 $0 - $9를 리용하여 직접 얻을수 있다.

$0　　지령이름
$1　　첫번째 인수
$2　　두번째 인수
$3
　　∫
$9　　아홉번째 인수

프로그람이 실행될 때 얼마나 많은 지령행인수들을 얻을수 있는가 하는것이 명백치 않을 때에는 다음과 같은 두개의 변수들을 리용할수 있다.

$#　　지령행인수들의 개수
$*　　공백으로 분리된 지령행인수들(지령이름은 포함하지 않는다.)

변수 $*은 흔히 여러개의 인수들을 가진 쉘스크립트지령행들을 처리하기 위하여 for 순환과 함께 리용된다.

이제 이미 앞에서 취급한 myll을 프로그람이 실행될 때 사용자가 규정해 주는 등록부의 내용들을 보여 주도록 변경하자. 그림 12-1에서는 변경된 myll을 보여 주고 있다.

```
#! /bin/ksh
# This is a simple shell program that takes one command line
# argument (a directory name) and then displays the full pathname
# of that directory before doing a long file listing (ll) on
# it.
#
# The script name is myll
cd $1
echo "Long listing of the `pwd` directory:"
echo
ls -l
```

그림 12-1. myll 쉘프로그람

만일 myll을 등록부이름과 함께 실행시키면 스크립트등록부를 변경하고 경로이름을 포함하는 통보문을 출력한 다음 ls -l 지령을 수행시킨다. myll 프로그람에 있는 cd는 스

크립트의 작업등록부만을 변경시키며 myll 을 실행시킨 쉘의 작업등록부에는 영향을 주지 않는다.

```
martyp $ myll /tmp
Long listing of the /tmp directory:

total 2384
-rw-------   1  root    sys     265228  Feb  22  15:21  dtdbcache_:0
-rw-r--r--   1  root    sys      70829  Feb  23  10:44  g
-rw-r--r--   1  root    sys      13642  Feb  23  10:48  i
-rw-rw-rw-   1  root    root     14071  May  24  06:10  license_log
-rwxr-xr-x   1  chrisb  users      317  Apr  20  17:38  ls
-rw-rw-r--   1  root    sys       4441  Mar  25  14:37  mwaps7454
-rw-rw-rw-   1  anne    users     4341  May  20  13:56  ps23974
-rw-r--r--   1  rodt    users     4218  Apr  14  11:17  ps3358
-rw-r--r--   1  rodt    users     4763  Feb  24  07:23  ps6465
-rw-rw-r--   1  root    sys       4446  Mar  25  14:31  ps7036
-rw-rw-r--   1  root    sys       4442  Mar  25  14:35  ps7138
-rw-rw-r--   1  root    sys       4446  Mar  25  14:35  ps7215
-rw-rw-r--   1  root    sys       4498  Mar  25  14:36  ps7342
-rw-rw-r--   1  root    sys       4446  Mar  25  14:38  ps7622
-rw-rw-r--   1  root    sys       4615  Mar  25  15:30  ps7812
-rw-rw-r--   1  root    sys       5728  Feb  18  11:09  ps_data
-rw-r--r--   1  root    sys          0  Apr  26  10:50  sh20890.1
-rw-r--r--   1  root    sys          0  Apr  26  10:50  sh20891.1
-rw-r--r--   1  root    sys          0  Apr  26  10:51  sh20978.1
-rw-r--r--   1  root    sys          0  Apr  26  10:51  sh20979.1
-rw-r--r--   1  chrisb  users     5325  Mar  26  13:42  sman_9664
-rw-rw-r--   1  root    sys     295996  Mar   1  10:15  ups_data
drwx------   2  root    other       69  Mar   9  11:37  whatis.11526
drwx------   2  root    other       69  Mar   9  11:37  whatis.11686
drwx------   2  root    other       69  Mar   9  11:38  whatis.11853
drwx------   2  root    other       69  Mar   9  11:38  whatis.12014
-rw-r--r--   1  root    sys     354221  Feb  23  10:49  x
-rw-r--r--   1  chrisb  users        0  Feb  23  14:39  xx
martyp $
```

이 경우 myll 에 인수를 줄수 없지만 프로그람은 정확히 동작한다. 만일 그 어떤 지령행인수도 제공하지 않으면 $1 은 null 로 되며 따라서 지령행의 cd 뒤에는 아무것도 없게 된다. 이것은 cd 를 홈등록부에 가져 오고 거기에서 ll 을 수행시킨다.

한개이상의 인수들을 주면 첫 인수만이 리용되고 다른 인수들은 무시된다.

한개의 지령행인수를 리용하는 경우에 그것은 등록부이름이여야 한다. 그렇지 않으면 cd지령은 실패하며 스크립트는 "bad directory"라는 오유통보문을 내보내고 끝마친다.

보다 복잡한 실례는 ps지령의 새 방안을 만드는데 리용될수 있다. 아래에서는 프로쎄스관리를 돕는데 지령행인수와 지령치환을 리용하는 두개의 실례를 주었다.

그림 12-2 의 psg 쉘프로그람은 일정한 지령들 또는 사용자프로쎄스들을 찾기 위한 프로쎄스상태용지를 탐색하는데 리용된다. 이 실례들은 grep 를 리용한다. grep 는 탐색하고 있는 패턴을 포함하는 모든 행들을 찾는다.

```
#!/usr/bin/sh
# Program name: psg
# Usage: psg some_pattern
#
# This program searches through a process status (ps -ef)
# listing for a pattern given as the first command-line
# argument.
procs='ps -ef'                        # Get the process listing
head='echo "$procs" | line'           # Take off the first line (the headings)
echo "$head"                          # Write out the headings
echo "$procs" | grep -i $1 | grep -v $0 # Write out lines
    # containing $1 but not this program's command line
# Note that $procs MUST be quoted or the newlines in the ps
# -ef listing will be turned into spaces when echoed. $head
# must also be quoted to preserve any extra white space.
```

그림 12-2. psg 쉘프로그람

psg 를 실행시키면 다음의 결과를 얻는다. 이 실례에서 우리는 체계상에서 실행되는 모든 Korn쉘들을 살펴 보기로 한다.

```
martyp $ pag ksh
  UID     PID    PPID  C        STIME   TTY   TIME  CMD
  root    2954   2936  0     Feb   22     ?   0:01  /bin/ksh/usr/dt/bin/Xsession
  root    3002   2999  0     Feb   22 pts/2   0:01  -ksh -c   unset DT; DISPLg
  root    3067      1  0     Feb   22     ?   0:00  /bin/ksh /usr/dt/bin/sdtvo1checkm
 jnola   11516  11514  0     May   11 pts/3   0:01  -ksh
martyp   29291  29289  0 09:30:04  pts/4   0:02    -ksh
```

이 프로그람은 말단, 프로쎄스 ID, 선조프로쎄스 ID, 시작날자 그리고 ps 로부터 임의의 다른 정보를 찾을수 있다.

사용자로서는 정지되기를 바라는 프로쎄스들을 시동시킬수 있다. 실례로 화면에 나타나지 않는 응용프로그람을 시동시킬수 있다. 프로쎄스를 psg 로 지명할수 있으며 정지시킬 권리가 있는 프로쎄스를 정지시키기 위하여 다음번 프로그람을 리용할수 있다.

그림 12-3 에 있는 gkill 쉘프로그람은 ps -ef 에서 패턴을 탐색하고(psg 와 같이) 렬거된 프로쎄스들을 제거한다. 아래의 실례에서는 cut 지령을 리용한다. cut 지령은 보류하려는 렬들의 범위를 규정할수 있게 한다.

```
#!/usr/bin/sh
# Program name: gkill
# Usage: gkill some_pattern
# This program will find all processes that contain the
# pattern specified as the first command line argument then
# kills those processes.
# get the process listing
procs='ps -ef'
echo "The following processes will be killed:"
# Here we list the processes to kill. We don't kill this
# process
echo "$procs" | grep -i $1 | grep -v $0
# Allow the user a chance to cancel.
echo "\nPress Return to continue Ctrl-C to exit"
# If the user presses Ctrl-C the program will exit.
# Otherwise this read waits for the next return character and
# continue.
read junk
# find the pattern and cut out the pid field
pids='echo "$procs" | grep -i $1 | grep -v $0 | cut -c9-15'
# kill the processes
kill $pids
```

그림 12-3. gkill 쉘프로그람

만일 지령행인수를 주지 않으면 grep 는 오유를 발생시키며 프로그람은 계속 수행된다. 다음 부분에서는 $1 이 설정되여 있는가를 어떻게 검사하며 그렇지 않은 경우에는 어떻게 지워 버리겠는가를 고찰한다.

이제 한 프로쎄스를 배경에서 시동시키고 그 프로쎄스번호를 지명하는데 psg 프로그람을 리용하며 그다음 gkill 을 리용하여 그 프로쎄스를 정지시키자.

```
martyp $ find / -name .c > cprogs 2>&1 &
[1]      29683
martyp $ psg find
```

```
    UID     PID     PPID    C       STIME               TTY     TIME  CMD
    martyp  29683   29579   7       13:54:19    pts/4   0:02  find / -name .c
martyp $ gkill 29683
The following processes will be killed:
    martyp  29683   29579  10       13:54:19    pts/4   0:03  find / -name .c
Press Return to continue.Ctrl-C to exit

martyp $
```

find 지령을 실행시키면 프로쎄스번호를 알게 된다. 잘못된 프로쎄스를 제거하지 않으려면 psg 와 함께 실행되고 있는 모든 find 프로그람들을 검사한다. 제거하려는 프로쎄스의 번호가 정확하다면 psg 를 리용하여 그 프로쎄스를 제거한다.

이 부분의 쉘프로그람들은 비록 단순하지만 많은 중요한 쉘프로그람작성기술들을 리용하고 있다. 다음으로 쉘프로그람작성의 가장 중요한 개념들인 검사와 갈래에 대하여 고찰한다.

검사와 갈래

결심채택은 쉘의 강력한 특성들중의 하나이다. 조건을 검사하고 그 조건을 취급하는 코드부분에로 갈라 지는 방법은 두가지이다.

실례로 사용자에게 질문을 제기하고 대답이 yes 인가 no 인가를 검사할수 있으며 조작하기전에 파일이 존재하는가를 검사할수 있다. 이러한 경우들에 if 지령을 리용할수 있다.

아래에 if지령의 매 부분을 설명하는 작은 쉘스크립트토막이 있다.

```
echo "Continue? \c"
read  ans
if [ "$ans" = "n" ]
then
    echo "Goodbye"
    exit
fi
```

echo 와 read 는 입력재촉문과 응답을 제공한다. if 명령문은 다음번지령을 수행하고 만일 그것이 성공하면 then 과 fi(if 를 거꾸로 쓴것)사이에 있는 지령들을 수행한다.

echo 지령에 있는 \c 는 echo 가 표준적으로 발생시키는 줄바꾸기를 억제한다. 이 작용은 유표를 "Continue?"입력재촉문의 바로 뒤에 머물러 있게 한다. 이것은 사용자로부터의 입력을 요구할 때 흔히 리용된다.

test 지령은 if 명령문과 자주 함께 리용된다. ["$ans" = "n"]은 test 지령이다. 이 지령으로 파일형, 기호렬, 수리론리적인 검사 등을 진행하는데 만일 조건이 진실이면 검사는 성공한것이다.

test 지령의 문장형식은 [ ]의 주위에 공백을 요구하는데 그렇지 않으면 프로그람이 실행될 때 문장론오유가 발생된다. 또한 응답변수 $ans 주위에 웃두점을 쳐야 한다. 만일 사용자가 입력재촉문에 다른 기호들은 없이 [[RETURN]]만을 써넣으면 $ans 의 값은 null 로 된다. $ans 주위에 웃두점을 치지 않으면 $ans 의 값이 test 지령안에 치환되였을 때처럼 보인다. 즉

```
[ = "n" ]
```

프로그람을 실행시키면 이것은 "test: argument expected"오유를 발생시킨다. 이것은 가장 흔히 나타나는 실수이며 이 오유에 맞다들면 test 지령에 null 값을 가진 변수가 있는가를 보아야 한다.

다른 형식의 if 지령이 있다.

```
if  [ ]                 # if  some  condition  is  true
then
                        # do  something
else
                        # otherwise  do  this
fi
```

이 지령은 조건이 옳으면 then 다음의 지령을 수행하고 그렇지 않으면 else 다음의 지령을 수행하게 한다.

test 지령이 검사할수 있는 조건들은 많다. 표 12-1 에서는 검사할수 있는 조건들중의 일부를 보여 주고 있다.

표 12-1　　　지령 test 의 조건들

기호렬검사들 :	설　명
["$a" = = "string"]	$a 가 "string"과 같으면 진실이다.
["$a" ! = "string"]	$a 가 "string"과 같지 않으면 진실이다.
[-z "$a"]	$a 가 null 이면 진실이다.
[-n "$a"]	$a 가 null 이 아니면 진실이다.
수값검사들 :	**설　명**
[$x -eq 1]	$x 가 1 과 같으면 진실이다 .
[$x -ne 1]	$x 가 1 과 같지 않으면 진실이다.
[$x -lt 1]	$x 가 1 보다 작으면 진실이다.
[$x -gt 1]	$x 가 1 보다 크면 진실이다.
[$x -le 1]	$x 가 1 보다 작거나 같으면 진실이다.
[$x -ge 1]	$x 가 1 보다 크거나 같으면 진실이다.

(표계속)

파일검사들 :	설　명
[-d $file]	$file이 등록부이면 진실이다.
[-f $file]	$file이 파일이면 진실이다.
[-s $file]	$file이 0byte이상이면 진실이다.
[-r $file]	$file이 읽기가능하면 진실이다.
[-w $file]	$file 이 쓰기가능하면 진실이다.
[-x $file]	$file이 실행가능하면 진실이다.

여러개의 검사들을 론리적으로 결합할수 있는데 -a를 리용하면 "AND", -o를 리용하면 "OR", !를 리용하면 "NOT"로 된다. 즉

["$interactive" = "TRUE" -o -d $file]

이 test명령문은 변수 $interactive가 진실이거나 $file이 등록부일 때 진실이다.

다음 실례들에서는 이 명령문을 리용한 프로그람들을 보여 주고 있다.

프로그람이 처리를 하기전에 정확히 한개의 지령행인수가 있다는것을 보여 줄수 있게 앞에서 본 프로그람 gkill을 확장하자.

이것을 위하여 수값검사와 지령행인수들의 개수를 표현하는 $#변수를 우에서 주어진 gkill실례의 앞에 삽입하자.

```
# If we don't have exactly one command-line argument,
# write an error and exit.
if [ $# -ne 1]
then
    echo "Usage: $0 pattern"
    echo "Some pattern matching the processes to kill""
    echo "must be specified"
    exit 1   # Exit 1 terminates the program and tells the
             # calling shell that we had an error.
fi
```

gkill프로그람을 일부 다르게 확장하여

- 사용자가 kill지령과 함께 리용할 신호를 규정하게 할수 있다. 실례로 지령 gkill -9 ralph 는 ralph의 모든 프로세스들을 찾고 그것들을 지령 kill -9로 제거한다.
- 규정된 패턴을 리용해서는 제거하려는 프로세스를 찾을수 없는 경우에 해당한 통보문이 인쇄되도록 할수 있다.

이러한 형태의 지령행검사는 psg 프로그람에 쉽게 적용되여 탐색해야 할 패턴을 표현하는 인수가 정확히 한개가 되도록 한다.

사용자입력을 읽을 때 사용자가 값을 입력하였는가를 검사하려고 할수 있다. 만일 값을 입력하지 않았다면 합리적인 기정값을 마련하여야 한다. 이것을 변수조작기로 쉽게 할수 있다.

아래 실례에서는 사용자로부터 대답("ans")을 읽고 그 값을 if 지령을 리용하여 검사한다.

```
echo "Do you really want to remove all of your files? \c"
read ans
if [ ${ans:-n} = = y ]
then
   rm -rf *
fi
```

${ans:-n}명령문은 $ans 의 값을 검사한다. $ans 에 값이 있으면 그것을 지령행에서 리용한다. 만일 사용자가 입력재촉문에 대하여 단순히 [[RETURN]]을 눌렀다면 $ans 는 null 일것이다. 이 경우에 ${ans:-n}은 비교할 때 n 으로 평가된다. 한개 작은 명령문에서 그것은 "사용자가 대답을 주지 않으면 n 을 의미하는것이라고 가정하라"는것을 말해 준다. 자주 리용되는 다른 조작은

```
${var:=default}
```

이다. 이것은 var 가 설정된 경우에 그 값을 돌려 준다. var 가 설정되여 있지 않으면 기정값을 돌려 주고 그 기정값을 var 에 대입하여 후에 리용되도록 한다.

case 명령문에 의한 판단

case 명령문은 쉘프로그람들에서 조건을 검사하고 결심을 채택하는 또 다른 방법이다. 이 명령문은 지령행인수들에 있는 일정한 패턴들을 검사하는데 흔히 리용된다. 실례로 첫번째 지령행인수가 선택항목(a-로 시작되는)인가를 결정하려는 경우에 case 명령문으로 쉽게 할수 있다. case 명령문은 또한 각이한 사용자입력(사용자에게 차림표로부터 한 항목을 선택하라고 묻는것과 같은)에 응답하는데도 리용된다.

명령문 case 는 문장형식이 가장 복잡한 쉘명령문들중의 하나이다. 즉

```
case pattern_to_match in
        patternl) cmdA
                  cmdB
                  ; ;
        pattern2)  cmdC
```

```
                ; ;
              …
        *) cmdZ
                ; ;
esac
```

이 형식에서 pattern_to_match 는 검사하고 있는 (지령행인수 또는 사용자응답과 같은) 쉘변수이다. 만일 pattern_to_match 가 pattern1 과 맞는다면 지령 cmdA 와 cmdB 가 실행된다. ; ;는 한 패턴의 지령렬을 다음 패턴과 분리시킨다. 모든 경우에 ; ;에 도달하면 프로그람은 esac(case 를 거꾸로 쓴것)에로 이행한다.

만일 pattern_to_match 가 pattern2 와 맞으면 cmdC 가 실행되고 case 명령문의 끝이 esac 에로 이행한다.

*은 pattern_to_match 가 그 어느것과도 맞지 않을 때 cmdZ 를 실행하도록 하기 위한것이다. 사용자가 무효한 입력을 진행한 경우에 처리해야 할 기정작용을 마련해 놓는것이 중요하다.

패턴정합을 보다 더 원만히 진행하려면 파일이름생성기호들(*, [ ], ?)을 리용하여야 한다. 론리합 "OR"를 의미하는 기호(|)를 리용하여 한개 행에서 여러 패턴에 대한 검사를 진행할수 있다. 한가지 실례를 고찰하자.

```
echo "Do you want to continue? (y/n) \c"
read ans
case $ans in
        y | Y) echo "Continuing"
              …
              ; ;
        n | N) echo "Done, Goodbye"
              exit
              ; ;
        *) echo "Invalid input"
              ; ;
esac
```

다음 실례에서는 $1(첫 지령행인수)이 유효한 선택항목인가를 알아내는 검사를 진행한다.

```
case $1 in
        -1 | -d) # Perform a listing
              echo "All files in $HOME:\n"
              11 -R $HOME | more
```

```
            ; ;
        -i) # -i means set to an interactive flag to true
         interactive="TRUE"
            ; ;
        *) # Invalid input
         echo "$0: $1 is an invalid option"
         exit 1
         ; ;
esac
```

case 명령문은 후에 이 장의 다른 실례들에서도 리용된다.

순환

어떤 동작을 여러번 반복하여 수행시킬 필요가 있을수도 있다. 쉘에서는 이것을 다음과 같은 두가지 방법으로 실현한다.

① for 순환은 항목들의 목록을 정하고 그 목록의 매개 항목에 대하여 순환내부의 지령들을 한번씩 수행한다.

② while 순환은 검사조건이 진실이면 순환내부의 지령들을 실행하고 다시 순환할수 있겠는가를 판정하기 위하여 검사조건을 다시 검사한다.

for 순환의 기본형식은 다음과 같다.

```
for var in list_of_items
do
        cmdA
        cmdB
        cmdC
done
```

순환이 시작될 때 var 의 값은 순환하려고 하는 list_of_items 안의 첫 단어로 설정된다. 그다음 do 와 done 사이에 있는 3 개 지령들이 수행된다. 프로그람이 done 명령문에 도달한 다음 순환의 꼭대기로 돌아 가며 var 에는 그 목록안의 다음번 항목이 대입된다. 그리고 지령들을 다시 수행한다. 순환이 끝났을 때 프로그람은 done 명령문뒤의 다음번 수행가능한 명령문에서 계속된다.

list_of_items 는 공백으로 분리된 단어들의 목록일수 있다. 목록을 구성하기 위하여 단어들을 입력하거나 변수 또는 지령치환을 리용할수 있다. 실례로 새로운 .kshrc 파일을

여러 사용자들의 홈등록부내부에 복사하려고 한다고 할 때 이것을 for 순환으로 쉽게 할 수 있다. 즉

```
for name in ralph norton alice edith archie
do
        echo $name
        cp /tmp/.kshrc.new /users/$name/.kshrc
done
```

이 실례는 일정한 파일들을 rcp 지령을 리용하여 여러 장치들에 복사하고 그것들이 거기에 있는가를 remsh 지령을 사용하여 확인하도록 확장할수 있다.

```
for host in neptune jupiter mars earth sun
do
        echo $host
        rcp /etc/passwd /etc/hosts $host:/etc
        rcp /.profile $host:/.profile
        remsh $host 11 /etc/passwd /etc/hosts /.profile
done
```

또한 list_of_items 를 생성하는데 지령치환을 리용하여 한 등록부에 있는 파일들의 목록을 처리할수도 있다.

```
for file in 'ls'
do
        if [ -r $file ]
        then
                echo "$file is readable
        fi
done
```

for file in *도 동일한 과제를 수행한다.

순환하려고 하는 항목들의 목록이 크고 그것들을 지령행에 쓰고 싶지 않으면 그것들을 대신 파일에 넣을수 있다. 그러면 cat 지령과 지령치환을 리용하여 list_of_items 을 생성할수 있다.

```
for i in `cat important_files`
do
  # do something with each of the files listed in the
```

```
    # important_files file.
done
```

하지만 for 순환은 지령행인수들의 목록($*)들을 처리하는데 자주 리용된다.

```
for name in $*
do
    if      [ ! -f $name -a ! -d $name ]
    then
            echo "$name is not a valid file or directory name"
    else
            # do something with the file or directory
    fi
done
```

뒤의 trash 프로그람은 지령행인수를 류사한 방법으로 처리하는 for 순환을 포함하고 있다.

while 순환

while 순환의 형식은 다음과 같다.

```
while cmd1
do
        cmdA
        cmdB
        cmdC
done
```

cmd1 이 먼저 수행된다. 만일 그것이 성공적으로 수행되면 do 와 done 명령문들사이에 있는 지령들이 수행된다. cmd1 이 다시 성공적으로 수행되면 순환속의 지령들이 다시 수행된다. cmd1 이 실패할 때 프로그람은 done 명령문뒤에 놓인 다음번 수행가능한 명령문을 계속 수행한다.

cmd1 의 위치에서 실행되는 지령은 대부분 test 지령이다. if 부분에서 서술된 론리적인 검사들도 수행될수 있다. 검사가 성공하면(진실이면) 순환속의 지령들이 수행되고 스크립트는 조건을 다시 검사한다. while 순환은 순환하려고 하는 회수가 고정되였거나 또는 어떤 조건이 만족될 때까지 무엇인가를 하려고 하는 경우에 리용될수 있다.

while 을 리용하는 실례를 보기로 하자. netstat 지령을 매 30 초에 한번씩 초등적인 (primary) LAN 대면부의 출력을 보면서 열번 수행시키려고 한다. 현재 실행하고 있는

Solaris 체계의 경우에 이것은 le0 이다. Solaris 체계를 리용하고 있지 않으면 **초등적인 LAN 대면부**는 각이한 이름을 가질수 있다.

우선 하나의 간단한 프로그람을 보기로 하자.

이 프로그람은 netstat 를 30 초에 한번씩 열번 리용하면서 초등적인 LAN 대면부(le0) 통계를 현시하는데 그것은 lan0 이거나 리용하고 있는 UNIX 변종에 따라 어떤 다른 이름일수 있다.

```
# !/bin/ksh
i = 1
while [ $i -le 10 ]
do
        print $i
        netstat -i | grep le0
        sleep  30
        let  i=i+1
done
```

매번 순환에서 1 을 더하는 방법으로 i 를 간단히 증가시킨다.

순환프로쎄스에는 매번 i 의 값이(le) 10 보다 작은가 아니면 같은가를 결정하기 위한 평가를 한다. 만일 그렇다면 netstat -i 를 실행시킨다.

net1 이라고 부르는 이 프로그람을 실행시키기전에 그 출력을 보기 위하여 netstat -i 를 실행시킨 다음 net1 을 실행시킨다. netstat 지령은 UNIX 변종들속에서 크게 변하기때문에 출력이 크게 차이난다는것을 알아야 한다.

```
martyp $ netstat -i
Name   Mtu   Net/Dest   Address    Ipkts       Ierrs   Opkts      Oerrs   Collis  Queue
lo0    8232  loopback   localhost  18927030      0     18927030     0      0        0
le0    1500  sunsys     sunsys     310417964     0     17193381   52064  7573173
sunsys:/home/martyp/shellprogs
martyp $ netl
1
le0 1500  sunsys    sunsys   310418018  0    17193388  52064  7573173
2
le0 1500  sunsys    sunsys   310418738  0    17193395  52064  7573173
3
le0 1500  sunsys    sunsys   310419579  0    17193401  52064  7573173
4
le0 1500  sunsys    sunsys   310420295  0    17193405  52064  7573173
5
```

```
le0 1500  sunsys      sunsys   310421099  0     17193446  52064  7573173
6
le0 1500  sunsys      sunsys   310421786  0     17193455  52064  7573173
7
le0 1500  sunsys      sunsys   310422425  0     17193462  52064  7573173
8
le0 1500  sunsys      sunsys   310423089  0     17193467  52064  7573173
9
le0 1500  sunsys      sunsys   310423749  0     17193471  52064  7573173
10
le0 1500  sunsys      sunsys   310424507  0     17193478  52064  7573173
sunsys:/home/martyp/shellprogs
martyp  $
```

net1 은 sunsys 라고 부르는 체계에서 매번 30 초마다 le0 에 대하여 요구하는 출력을 생성한다. 또한 i 의 값을 알맞게 증가시키고 있다는것을 볼수 있도록 매 순환에서 i 의 값을 인쇄한다.

HP-UX 11i 체계에서 netstat 지령은 정보를 적게 제공하도록 변경되여 있다. 충돌과 오유들에 대한 자세한 정보를 얻으려면 lanadmin 지령을 리용할수 있다. 다음의 스크립트는 11i 체계우에서 망대면부카드들의 목록을 자동적으로 생성시키기 위하여 lanscan 을 실행시킨 다음 매 카드에 대하여 lanadmin 출력을 생성한다.

```
#! /bin/sh
#@(#) $Header: $
#@(#) Description: Script to create network stats output with lanadmin
#@(#) $Source: $
#@(#) $Locker: $
#@(#) $Log: $
#@(#)
# INPUT:
# OUTPUT:
# FILES ACCESSED:
# NOTES:
# AUTHOR:
# REVISION HISTORY:
# Setup path
PATH=/bin:/usr/bin:/usr/sbin/:/usr/local/bin/;export PATH
#
# Determine OS Revision level
```

```
# run lanscan to determine network cards
#
OS_VERSION=$(uname -r | awk -F. '{print $21}')
if [ $OS_VERSION -lt 10]
then
    COMMAND1=/bin/landiag
else
    COMMAND1=/usr/sbin/lanadmin
fi
if  [ -x $COMMAND1]
then
    COMMAND2=/usr/sbin/lanscan
    if [ -x $COMMAND2]
    then
              if [ $OS_VERSION -lt 10]
              then
                for CARD in `ls -1 /dev/lan* | awk -F"/" '{print $31}' `
                do
                        echo " "
$COMMAND1 -t <<- EOF 2> /dev/null | egrep " = "
lan
name $CARD
display
quit
EOF
    done
#
# usefield 3 of lanscan output to obtain card instance used for ppa below
#
    else
      for NMID in `$COMMAND2 | grep 0x | awk '{print $3}' `
      do
              echo " "
$COMMAND1 -t <<- EOF 2> /dev/null | egrep " = "
lan
ppa $NMID
display
```

```
quit
EOF
    done
  fi
else
  print
  print
  print "The command \"$COMMAND2\" is not executable"
  print "No data from \"$COMMAND1\" will be collected"
  print
  print
fi
else
  print
  print
  print "The command \"$COMMAND1\" is not executable"
  print
  print
fi
```

lanscan 의 세번째 마당은 lanadmin 에 ppa 정보를 **카드구체례**(Card Instance)로 제공하기 위하여 리용된다. lanadmin 출력은 그 카드구체례에 대하여 생성되게 된다. 다음의것은 lanscan의 출력인데 다섯개의 망카드가 있는 체계에서 그 스크립트를 실행시킨 결과이다.

```
# lanscan

Hardware  Station           Crd  Hdw    Net-Interface  NM  MAC   HP-DLPI  DLPI
Path      Address           In#  State  NamePPA        ID  Type  Support  Mjr#
0/0/0/0      0x001083FEDCB7  0    UP    lan0  snap0    1   ETHER  Yes     119
0/4/0/0/4/0  0x001083F72ED0  1    UP    lan1  snap1    2   ETHER  Yes     119
0/4/0/0/5/0  0x001083F72E9B  2    UP    lan2  snap2    3   ETHER  Yes     119
0/4/0/0/6/0  0x001083F77E22  3    UP    lan3  snap3    4   ETHER  Yes     119
0/4/0/0/7/0  0x001083F71E59  4    UP    lan4  snap4    5   ETHER  Yes     119
```

```
# networkscript.sh

PPA Number            = 0
Description           = lan0 Hewlett-Packard 10/100 TX Half-Duplex TT =1500
Type（value）          = ethernet-csmacd(6)
```

```
MTU Size                        = 1500
Speed                           = 10000000
Station Address                 = 0x1083fedcb7
Administration Status (value)   = up(l)
Operation Status (value)        = down(2)
Last Change                     = 10440
Inbound Octets                  = 0
Inbound Unicast Packets         = 0
Inbound Non-Unicast Packets     = 0
Inbound Discards                = 0
Inbound Errors                  = 0
Inbound Unknown Protocols       = 0
Outbound Octets                 = 820
Outbound Unicast Packets        = 20
Outbound Non-Unicast Packets    = 0
Outbound Discards               = 0
Outbound Errors                 = 0
Outbound Queue Length           = 0
Specific                        = 655367
Index                           = 1
Alignment Errors                = 0
FCS Errors                      = 0
Single Collision Frames         = 0
Multiple Collision Frames       = 0
Deferred Transmissions          = 0
Late Collisions                 = 0
Excessive Collisions            = 0
Internal MAC Transmit Errors    = 0
Carrier Sense Errors            = 0
Frames Too Long                 = 0
Internal MAC Receive Errors     = 0
PPA Number                      = 1
Description                     = lan1 Hewlett-Packard 10/100 TX Full-Duplex TT=1500
Type (value)                    = ethernet-csmacd(6)
MTU Size                        = 1500
Speed                           = 100000000
Station Address                 =  0x1083f72ed0
Administration Status (value)   = up(l)
Operation Status (value)        = up(l)
```

```
Last Change                      = 11018
Inbound Octets                   = 2778542151
Inbound Unicast Packets          = 10008640
Inbound Non-Unicast Packets      = 14480929
Inbound Discards                 = 0
Inbound Errors                   = 0
Inbound Unknown Protocols        = 12443000
Outbound Octets                  = 3811379313
Outbound Unicast Packets         = 18378160
Outbound Non-Unicast Packets     = 50019
Outbound Discards                = 0
Outbound Errors                  = 0
Outbound Queue Length            = 0
Specific                         = 655367
Index                            = 2
Alignment Errors                 = 0
FCS Errors                       = 0
Single Collision Frames          = 0
Multiple Collision Frames        = 0
Deferred Transmissions           = 0
Late Collisions                  = 0
Excessive Collisions             = 0
Internal MAC Transmit Errors     = 0
Carrier Sense Errors             = 0
Frames Too Long                  = 0
Internal MAC Receive Errors      = 0

PPA Number                       = 2
Description                      = lan2 Hewlett-Packard 10/100 TX Full-Duplex TT=1500
Type（value）                    = ethernet-csmacd（6）
MTU Size                         = 1500
Speed                            = 100000000
Station Address                  = 0x1083f72e9b
Administration Status（value）   = up（1）
Operation Status（value）        = up（1）
Last Change                      = 12223
Inbound Octets                   = 1053052283
Inbound Unicast Packets          = 660
Inbound Non-Vnimt Packets        =14479097
```

Inbound Discards = 0
Inbound Errors = 0
Inbound Unknown Protocols = 12442909
Outbound Octets = 5802138
Outbound Unicast Packets = 43576
Outbound Non-Unicast Packets = 43065
Outbound Discards = 0
Outbound Errors = 0
Outbound Queue Length = 0
Specific = 655367
Index = 3
Alignment Errors = 0
FCS Errors = 0
Single Collision Frames = 0
Multiple Collision Frames = 0
Deferred Transmissions = 0
Late Collisions = 0
Excessive Collisions = 0
Internal MAC Transmit Errors = 0
Carrier Sense Errors = 0
Frames Too Long = 0
Internal MAC Receive Errors = 0

PPA Number = 3
Description = lan3 Hewlett-Packard 10/100 TX Full-Duplex TT=1500
Type (value) = ethernet-csmacd(6)
MTU Size = 1500
Speed = 100000000
Station Address = 0x1083f77e22
Administration Status (value) = up(1)
Operation Status (value) = up(1)
Last Change = 13428
Inbound Octets = 943616591
Inbound Unicast Packets = 9064639
Inbound Non-Unicast Packets = 765175
Inbound Discards = 0
Inbound Errors = 0
Inbound Unknown Protocols = 39
Outbound Octets = 6454687

Outbound Unicast Packets = 58769
Outbound Non-Unicast Packets = 43040
Outbound Discards = 0
Outbound Errors = 0
Outbound Queue Length = 0
Specific = 655367
Index = 4
Alignment Errors = 0
FCS Errors = 0
Single Collision Frames = 0
Multiple Collision Frames = 0
Deferred Transmissions = 0
Late Collisions = 0
Excessive Collisions = 0
Internal MAC Transmit Errors = 0
Carrier Sense Errors = 0
Frames Too Long - = 0
Internal MAC Receive Errors = 0

PPA Number = 4
Description = lan4 Hewlett-Packard 10/100 TX Full-Duplex TT=1500
Type (value) = ethernet-csmacd(6)
MTU Size = 1500
Speed = 100000000
Station Address = 0x1083f71e59
Administration Status (value) = up(1)
Operation Status (value) = up(1)
Last Change = 14633
Inbound Octets = 249984023
Inbound Unicast Packets = 2628160
Inbound Non-Unicast Packets = 765196
Inbound Discards = 0
Inbound Errors = 0
Inbound Unknown Protocols = 49
Outbound Octets = 3886863362
Outbound Unicast Packets = 10894938
Outbound Non-Unicast Packets = 425625
Outbound Discards = 0
Outbound Errors = 0

```
Outbound Queue Length          = 0
Specific                       = 655367
Index                          = 5
Alignment Errors               = 0
FCS Errors                     = 0
Single Collision Frames        = 0
Multiple Collision Frames      = 0
Deferred Transmissions         = 0
Late Collisions                = 0
Excessive Collisions           = 0
Internal MAC Transmit Errors   = 0
Carrier Sense Errors           = 0
Frames Too Long                = 0
Internal MAC Receive Errors    = 0
```

lanscan 을 실행하기직전에 스크립트는 ppa 번호를 출력하므로 lanscan 출력이 생성되게 될 망대면부카드를 알게 된다.

우의 실례에 있는 다섯개의 모든 망카드에 대하여서는 매 lanadmin 출력의 마지막에서 보여 준바와 같이 충돌과 오유들은 없다. 그러나 그 마당들은 lanadmin 에 의해 생성되였다.

while 순환은 지령행인수들을 그 지령행인수들의 개수와 shift 지령을 리용하여 한번에 처리하는데 리용될수도 있다.

```
while [ $# -ne 0 ]
do
    case $1 in
    -*) # $1 must be an option because it starts with-
          # Add it to the list of options:
          opts="$opts $1"
           ;  ;
    *) # $1 must be an argument. Add it to the list of
         # command-line arguments:
         args="$args $1"
          ;  ;
   esac
   shift
done
```

shift 지령은 $*에 남아 있는 인수들을 왼쪽으로 한자리 밀고 $#을 감소시킨다. 첫

번째 인수($1)였던것은 영원히 없어 지며 $2 에 있던것은 $1 에 들어 간다. 지령행인수들을 밀기하는 프로쎄스에 $#은 $*에 남아 있는 인수들의 개수를 정확히 반영하도록 감소된다.

사용자가 프로그람을 정지시킬 때까지 또는 어떤 정지조건을 만날 때까지 어떤 지령들을 실행하려고 할수 있는데 이것을 위한 가장 좋은 방법은 무한 while 순환이다.

실례로 사용자들에게 어떤 입력을 요구하고 유효한 입력을 할 때까지 계속 요구하려고 한다고 하자. 이때 프로그람을 다음과 같이 작성할수 있다.

```
while true
do
    # prompt users and get their response
    echo "Enter yes or no: \c"
    read ans
    # Check whether the response is valid
    if [ "$ans" == "yes" -o "sans" == "no" ]
    then
    # If it is valid, stop the looping
    break
else
    # otherwise print an error message and try it again
    # from the top of the loop
    echo "Invalid input, try again!\n"
    fi
done
# Now that we have valid input, we can process the user's
# request
 ⟆
```

true 는 항상 성과적으로 실행되는 특수한 지령이다. 순환은 사용자가 프로그람을 정지시킬 때까지 또는 순환에서 break 지령이 실행될 때까지 멈추지 않는다. break 지령은 순환을 정지시킨다.

쉘함수

쉘프로그람을 작성할 때 프로그람의 여러 곳에서 나타나는 일정한 지령모임들이 있는것을 볼수 있다. 실례로 한 스크립트에서 여러번 사용자입력을 검사하고 그 입력이 무효하면 적당한 통보문을 내보낼수 있다. 프로그람에 동일한 코드렬을 여러번 기입하거나 그것을 후에 변화시키는것은 지루한 일이다.

그대신 그 지령들을 하나의 쉘함수안에 넣을수 있다. 함수들은 스크립트안에서 리용

될수 있는 새로운 지령처럼 보이면서 작용한다. 간단한 쉘함수의 실례를 하나 고찰하자.

```
# This is a function that may be called from anywhere within
# the program. It displays a standard usage error message
# and then exits the program.
print_usage( )
{
    echo "Usage:"
    echo "To trash files: $0 [-i] files_to_trash..."
    echo "Display trashed files: $0 -d"
    echo "Remove all trashed files: $0 -rm"
    echo "Print this message: $0 -help"
    exit 1
}
```

print_usage 는 쉘프로그람에서 새로운 지령으로 된다. 그것을 이 스크립트안의 어디에서나 리용할수 있다. 쉘함수들은 자체의 위치적파라메터($1-$9, $#, $*)들을 가지므로 다른 지령들처럼 그것들을 인수로 전달할수 있다. $0 이 함수의 이름이 아니라 쉘프로그람의 이름을 표현하고 있다는것만이 다르다.

앞에서 우리는 인수들에 대하여 고찰하였다.

쉘프로그람의 이름을 입력할 때 변수 $1-$9 에 보관되여 있는 인수들을 넘겨 줄수 있다. 지령행의 첫 10 개 단어들은 쉘프로그람에서 특수한 변수 $0-$9 들을 리용하여 직접 얻을수 있다.

$0	지령이름
$1	첫번째 인수
$2	두번째 인수
	⸨
$9	아홉번째 인수

프로그람이 실행될 때 얼마나 많은 지령행인수들을 얻을수 있는가 하는것이 확실하지 않을 때에는 다음의 두개 변수들을 리용할수 있다.

$#	지령행인수들의 개수
$*	지령행인수들이 공백으로 분리된 목록(지령이름은 포함하지않는다.)

변수 $*은 임의의 개수의 인수들을 가진 쉘스크립트지령행들을 처리하는데 for 순환과 자주 함께 리용된다. 그림 12-4 에서 지적된 다음의 프로그람은 지금까지 취급한 모든 개념들을 련습하는 상당히 복잡한 프로그람을 보여 주고 있다. 이 프로그람은 파일들

을 본래 위치에서 제거하는 trash 프로그램인데 영원히 제거할 대신에 홈등록부에 옮겨 놓는다.

```
#! /bin/ksh
# for Bourne use /bin/sh
# Program name: trash
# Usage:
# To trash files:            trash [-i] file_names_to_trash ...
# Display trashed files:     trash -d
# Remove all trashed files: trash -rm
# Print a help message:      trash -help
# This program takes any number of directory or file name
# arguments. If the argument is a file it will be removed
# from its current place in the file system and placed in the
# user's trash directory ($HOME/.trash). If the argument is a
# directory name the program will ask if the User really
# wants to trash the whole directory.
#
# This program also takes an -i (interactive) option. Like
# the rm command, if the -i is the first argument on the
# command line, the program stops and asks if each file
# named in the remaining arguments should be trashed.
#
# The -d (display) option shows the contents of the
# user's trashed files.
#
# The -help option displays a usage message for the user.
# The -rm (remove) option interactively
# asks the user if each file or directory in the trash
# directory should be removed permanently.
#
# The -h, -d and -rm options may not be used with
# any other command line arguments.
# Possible extensions:
# - Enhance the -rm option to remove a list of files
# from the trash directory from the command line.
# - Create a program to be run by cron once nightly to empty
# everyone's trash directory.
#  This is a function that may be called from anywhere within
```

그림 12-4. trash 쉘프로그람

```
#   the program. It displays a standard usage error message
#   then exits the program.
print_usage（ ）
{
    echo "Usage:"
    echo "To trash files: $0 [-il file names to trash
    echo "Display trashed files:       $0 -d"
    echo "Remove all trashed files:  $0 -rm"
    echo "Print this message:                   $0 -help"
exit 1
}
# Make sure we have at least one command-line argument before
# we start.
if [ $# -lt 1]
then
    print_usage
fi
# If this flag is true then we need to do interactive
# processing.
interactive="FALSE"
# This is the name of the trash can.
trash-dir="$HOME/.trash"
# make sure the trash directory exists before we go any
# further.
if [ ! -d $trash_dir ]
then
    mkdir $trash_dir
fi
# Sort out the command-line arguments.
case $1 in
    -help）# Print a help message.
    print_usage
    ; ;
-d | rm）# a -d  or -rm were given
    # If it was not the only command-line argument
    # then display a usage message and then exit.
    if [ $# -ne 1]
    then
            print_usage
```

그림 12-4. trash 쉘프로그람(계속)

```
    fi
    # otherwise do the task requested.
    if [ $1 = = "-d"]
    then
            echo "The contents of $trash_dir:\n"
            ls -l -R $trash_dir | more
    else
            # remove all files from the trash directory
            rm -rf $trash_dir/*
            # get any dotfiles too
            rm -rf $trash_dir/.[!.]*
    fi
    # Now we can exit successfully.
    exit 0
    ; ;
-i) # If the first argument is -i ask about each file as it
    # is processed.
    interactive="TRUE"
    # Take -i off the command line so we know that the
    # rest of the arguments are file or directory names.
    shift
    ; ;
-i) # Check for an option we don't understand.
    echo "$1 is not a recognized option."
    print_usage
    ; ;
    esac
# Just for fun we'll keep a count of the files that were
# trashed.
count = 0
for file in $*
do
    # First make sure the file or directory to be renamed exists.
    # If it doesn't, add it to a list of bad files to be written
    # out later. Otherwise process it.
    if [ ! -f $file -a ! -d $file]
    then
            bad_files="$bad_files $file"
    else
```

그림 12-4. trash 쉘프로그람(계속)

```
# If we are in interactive mode ask for confirmation
# on each file. Otherwise ask about directories.
if [ "$interactive" = "TRUE" -o -d $file ]
then
   # Ask the user for confirmation (default answer is no).
   if [ -d $file]
   then
         echo "Do you want to trash the dir $file ? (y/n) n\b\c"
   else
         echo 'Do you really want to trash $file ? (y/n) n\b\c"
   fi
   read doit
   # If the user answered y then do the move.
   # Otherwise print a message that the file was not touched.
   if [ "$(doit:-n)ff = y]
   then
         mv -i $file $trash dir
         echo "$file was trashed to $trash-dir"
         let count=count+l
   # for Bourne use: count=`1expr $count + 1`
   else
         echo "$file was not trashed"
   fi
   else # We are not in interactive mode, so just do it.
   mv -i $file $trash_dir
   let count=count+l
   #for Bourne use: count=`expr $count + 1`
   fi
fi
done
echo "$0: trashed $count item(s)"
if [ -n "$bad_files"]
then
   echo "The following name(s) do not exist and \c"
   echo "could not be trashed:"
   echo "$bad_files"
fi
exit 0
```

그림 12-4. trash 셸프로그람(계속)

이제 trash 프로그람을 실행시켜 보자. 아래의 실레에서는 프로그람 trash 의 호출에 대하여 보여 주고 있다. 그 다음의 실레는 trash -help 호출을 보여 주고 그 다음의 실레는 junk 파일을 제거하는 trash -i junk 호출을 보여 주며 마지막실레는 trash 에 의하여 제거되고 /home/martyp/trash 등록부에 있는 파일들을 현시하는 trash -d 호출을 보여 준다.

```
martyp $ trash
Usage:
To trash files: trash [-i] file names_to_trash ...
Display trashed files:            trash-d
Remove all trashed files:    trash -rm
Print this message:          trash -help
martyp $ trash -help
Usage:
To trash files: trash [-i] file_names_to_trash ...
Display trashed files:            trash -d
Remove all trashed files:    trash -rm
Print this message:          trash -help
martyp $ trash -i junk
Do you really want to trash junk ? (y/n) y
mv: overwrite /home/martyp/.trash/junk (yes/no)? yes
junk was trashed to /home/martyp/.trash
trash: trashed 1 item(s)
martyp $ trash -d
The contents of /home/martyp/.trash:

/home/martyp/.trash:
total 1364
-rw-------      1    martyp  staff   684808  May 30 05:31   core
-rwxrwxr-x      1    martyp  staff   631     May 30 06:45   file1
-rwxrwxr-x      1    martyp  staff   45      May 31 06:04   junk
martyp $
```

파일 junk 를 제거하였을 때 trash 는 파일을 앞선 trash 에 의해 제거된 파일과 동일한 이름으로 덧쓰기를 하겠는가를 묻고 등록부 /home/martyp/trash 에 옮겨 놓는다.

이 프로그람들에는 이 장에서 지금까지 취급한 모든 개념들이 포함되여 있다. 이 프로그람을 잘 리해하면 이 프로그람에서 적용한 수법들을 다른 프로그람작성에서도 리용할수 있다. 이 프로그람을 Bourne 쉘에서 동작시키려면 설명문들을 교체하여야 한다. Korn 쉘과 Bourne 쉘은 매우 류사하므로 이 두 쉘들을 위한 프로그람을 작성할 때 동일한 수법들을 리용할수 있다.

쉘프로그람의 awk

awk는 매우 위력한 기호적인 프로그람작성언어이며 자료조작도구이다.

간단히 설명하면 awk는 입력행(표준입력 또는 파일로부터)에 있는 패턴들을 탐색한다. 규정된 패턴에 맞는 매 행에 대하여 awk는 그 행에 대한 매우 복잡한 처리를 수행한다. 맞는 입력행을 실지로 처리하는 코드는 쉘스크립트와 C프로그람사이의 결합이다.

grep, cut, paste의 결합으로 하여 매우 복잡한 자료조작과제들은 awk로 쉽게 완성할수 있다. awk는 프로그람작성언어이기때문에 쉘로는 보통 수행하기 힘든 수학적연산들을 수행하거나 입력검사를 매우 쉽게 할수 있다. 또한 류점수연산도 수행할수 있다.

awk프로그람의 기본형식은 다음과 같다.

```
awk `/pattern_to_match/ { program to run }' input_file_names
```

전체 프로그람을 웃반점안에 포함시킨다. 입력파일이름이 규정되지 않았으면 awk는 표준입력으로부터 읽는다.

pattern_to_match는 빗선기호(/)들사이에 들어 있다. 패턴을 정규식이라고 부른다. 일부 공통적인 정규식의 실례들은 간단하다.

수행하여야 할 프로그람을 C언어처럼 보이는 awk코드로 기입한다. 프로그람은 입력행이 pattern_to_match와 맞을 때 수행된다. /pattern_to_match/가 { }안에 있는 프로그람의 앞에 놓이지 않으면 그 프로그람은 매 입력행에 대하여 실행된다. awk는 입력행의 마당들을 가지고 작업한다. 마당들은 공백으로 분리된 단어들이다. awk패턴들과 프로그람들에 있는 마당들은 마당번호를 붙인 $로 참조된다. 실례로 입력행의 두번째 마당은 $2이다. 쉘프로그람들에서 awk지령을 리용하면 마당들($1, $2 등)은 쉘스크립트의 위치적파라메터와 혼돈되지 않는다. 왜냐하면 awk변수들은 웃반점안에 포함되며 쉘은 그것들을 무시하기 때문이다.

몇가지 실례들을 고찰하자.

```
who | awk `{ print $2 }'
```

이 간단한 실례는 체계에서 동작하고 있는 말단들만을 렬거한다. 체계에서 말단이름들은 who결과의 두번째 마당이다. 이 실례가 실행된 결과는 다음과 같다.

```
martyp $ who
thomfsu    console Feb 22 15:21  ( : 0 )
martyp     pts/3          May 31 06:03 (atlm02l6.atl.hp.com)
martyp $ who | awk '{print $2}'
console
pts/3
martyp $
```

이 출력은 체계에서 동작하는 말단들만을 보여 준다.

cut 도 역시 같은 일을 할수 있으나 아래에서 보는바와 같이 who 출력의 어느 렬에 말단이름이 놓이는가를 정확히 알아야 한다.

```
martyp  $  who
thomfsu     console  Feb  22  15:21   (:0)
martyp      pts/3              May  31  06:03  (atlm02l6.atl.hp.com)
martyp  $  who |   cut  -c12−20
console
pts/3
martyp  $
```

사용자 또는 말단이름이 행보다 길면 이 지령은 동작하지 않는다.

HP-UX 론리기록권관리쉘프로그람

"디스크첨가"를 취급한 8 장의 한개 항목에서는 다섯개의 **기본디스크**(Primary Disk)들과 다섯개의 **교대디스크**(Alternate Disk)들로 이루어 진 **기록권그룹**(Volume Group)을 수동으로 추가하는데 요구되는 HP-UX 걸음들을 보여 주고 있다. 이 수동절차는 10 개 기록권그룹들중 하나에 대하여 수행되였다. 남은 9 개의 기록권그룹들은 수동절차의 부분으로서 디스크들과 기록권그룹들의 이름을 변경시키면서 동일한 수동절차를 리용하여 첨가되여야 하였다.

이 걸음들을 수동으로 수행하는것은 오유를 많이 발생시킬수 있으므로 이것을 쉘프로그람으로 자동화하는것이 좋다.

이 절차를 자동화하기 위하여 XP256 상에서 1 차 및 교대경로로 리용되여야 할 물리적디스크들을 포함하는 파일을 고찰하자.

XP256 은 1 차조종기가 실패하는것을 교대조종기로 극복하는 능력을 가진 개선된 기억장치이다.

동일한 디스크모임들이 1 차조종기 및 교대조종기에 련결될수 있으나 그 디스크들은 두개의 각이한 장치파일모임으로 주어 진다. 첫번째 모임은 1 차조종기에 련결되였을 때 디스크들을 위한것이고 두번째 모임은 교대조종기에 련결되였을 때 동일한 디스크들을 위한것이다.

이것은 ServiceGuard 를 리용하는 경우에 맞다들리는것과 동일한 개념이다. 두개까지의 각이한 경로에 련결되는 디스크모임이 있으므로 디스크들을 그것이 1 차 또는 교대경로에 련결되는가에 따라 각이한 이름으로 정의하여야 한다. 아래에서는 다섯개씩 그룹화되여 있는 기본디스크들을 포함하는 파일 pri 를 보여 주고 있다.

```
c9t0d0   c9t0d1   c9t0d2   c8t0d0   c8t0d1
c7t0d0   c7t0d1   c7t0d2   c10t0d0  c10t0d1
c9t0d3   c9t0d4   c9t0d5   c8t0d3   c8t0d4
c7t0d3   c7t0d4   c7t0d5   c10t0d3  c10t0d4
c9t0d6   c9t0d7   c9t1d0   c8t0d6   c8t0d7
c7t0d6   c7t0d7   c7t1d0   c10t0d6  c10t0d7
c9t1d1   c9t1d2   c9t1d3   c8t1d1   c8t1d2
c7t1d1   c7t1d2   c7t1d3   c10t1d1  c10t1d2
c9t1d4   c9t1d5   c9t1d6   c8t1d4   c8t1d5
c7t1d4   c7t1d5   c7t1d6   c10t1d4  c10t1d5
```

여기에서 디스크들은 다섯개씩 그룹화되여 있다는것을 주목하여야 한다.

매 그룹은 하나의 기록권그룹을 구성한다. 여기에 있는 다섯개의 디스크로 된 총 10 개의 그룹은 vgu01-vgu10 까지 배치된다.

다섯개 디스크들의 교대그룹도 있을수 있다. 교대디스크들은 앞에서 서술한바와 같이 디스크조종기의 오유극복사건에서 리용되게 된다. 아래에서는 다섯개씩 그룹화되여 있는 교대디스크들의 목록을 포함하는 파일 alt 의 내용을 보여 주고 있다.

```
c8t8d0   c8t8d1   c8t8d2   c9t8d0   c9t8d1
c10t8d0  c10t8dl  c10t8d2  c7t8d0   c7t8d1
c8t8d3   c8t8d4   c8t8d5   c9t8d3   c9t8d4
c10t8d3  c10t8d4  c10t8d5  c7t8d3   c7t8d4
c8t8d6   c8t8d7   c8t9d0   c9t8d6   c9t8d7
c10t8d6  c10t8d7  c10t9d0  c7t8d6   c7t8d7
c8t9d1   c8t9d2   c8t9d3   c9t9d1   c9t9d2
c10t9d1  c10t9d2  c10t9d3  c7t9d1   c7t9d2
c8t9d4   c8t9d5   c8t9d6   c9t9d4   c9t9d5
c10t9d4  c10t9d5  c10t9d6  c7t9d4   c7t9d5
```

여기에 있는 총 10 개의 교대디스크그룹들은 vgu01- vgu10 까지의 기록권그룹들의 부분으로 된다.

XP256 상에서 설치된 1 차 및 교대디스크들을 리용하여 호스트체계상에서 알맞는 기록권을 설치할수 있다. 이 실례에서 호스트체계는 V-Class 체계이다.

이제 이 기록권그룹들중의 하나를 수동으로 창조하는 걸음들을 보기로 하자. 우선 pvcreate 지령으로 기록권그룹에 있는 매 디스크들에 대한 물리적기록권을 창조한다. 그다음 mkdir 로 기록권그룹에 대한 등록부를 창조하고 mknod 로 그 등록부안에 기록권그룹에 대한 장치특정의 파일을 창조한다. 이것은 10 개 기록권그룹들중 첫번째것에 대하여 요구되는 등록부와 특정의 파일들을 설치한다. 그다음 다섯개의 기본디스크들이 있게 될 기록권그룹을 vgcreate 를 리용하여 창조한다. 기록권그룹을 창조할 때 첫 디스크를

규정한 다음 그 그룹에 다른 디스크들을 vgextend 로 넣는다. 그 다음 그 그룹이 다섯개의 교대디스크들을 포함하도록 vgextend 로 확장시킨다.

마지막걸음은 총체적인 기록권그룹에 대한 하나의 론리기록권을 창조하는것이다. 하나의 기록권그룹안에서 여러개의 론리기록권을 창조할수도 있지만 여기에서 취급된 실례에서는 기록권그룹의 총 용량을 소비하는 한개의 론리기록권만을 고찰한다. 기록권그룹의 물리적크기는 8755 이다.

다음의 절차는 첫 기록권그룹을 창조하는 수동걸음들의 렬이다. 이것을 쉘프로그람으로 자동화하기로 한다.

```
# pvcreate /dev/rdsk/c9t0d0          # run for each of the 5 pri disks
# mkdir /dev/vgu01                   # make dir for first vol group
# mknod /dev/vgu01/group -c 64 0x01000
                                     # create special file with major and minor numbers
                                     shown
# vgcreate /dev/vgu01 /dev/dsk/c9t0d0
                                     # Place first disk in volume group
# vgextend /dev/vgu01 /dev/dsk/c9t0d1
                                     # extend volume with remaining four primary disks disks
# vgextend /dev/vgu01 /dev/dsk/c8t8d0
                                     # extend volume group to include five alternte disks
# lvcreate -1 8755 /dev/vgu01
                                     # creates lvol1 (lvol1 by default) that consumes all
                                     8755 extents
```

여기서 이 절차를 한개 디스크에 대하여서만 완성하였는데 이 기록권그룹에는 9 개의 추가적인 디스크들이 있다. 또한 9 개의 기록권그룹들이 더 있는데 이 절차는 그것들에 대하여서도 완성되여야 한다. 총 99 개의 추가적인 디스크들이 있는데 그 디스크들에 대하여 다양한 지령들이 수행되여야 한다. 입력할 때 많은 오유들이 생길수 있기때문에 입력을 자동화하는것이 좋다.

기본디스크모임과 교대디스크모임이 있기때문에 매 절차를 자동화하기 위한 짧은 프로그람을 작성하기로 한다. 아래의 프로그람에서는 매 디스크에 대한 물리적기록권과 기록권그룹을 창조하고 그안에 기본디스크들을 넣는 모든 걸음들을 수행하는 프로세스를 보여 주고 있다.

```
# i /bin/ksh
set -x              ;set tracing on
   vgnum=$l         ;first item-on each line is the volume group no.
   shift            ;shift to get to first disk

   for i in $*      ;run pvcreate for every disk name in first line
```

```
do
        pvcreate /dev/rdsk/$i
done
reada           ;pause program to view what has been run

mkdir /dev/vgu$vgnum                          ;mkdir for volume group
mknod /dev/vgu$vgnum/group c 0x$(vgnum)0000 ;mknod for volume group
vgcreate /dev/vgu$vgnum /dev/dsk/$l           ;vgcreate 1st disk in vg

shift           ;shift over to second disk
for i in $*     ;extend volume group to include remaining four disks
do
        vgextend /dev/vgu$vgnum /dev/dsk/$i
done

lvcreate -l 8755 /dev/vgu$vgnum               ;create single log vol for entire vg
```

실행추적(Execution Tracing)을 기동시키기 위하여 이 파일에 set -x 을 기입한다. 그러면 쉘프로그람의 **오유수정**(Debugging)을 시작하였을 때 실행되는 매행들을 볼수 있다. 실행되는 행은 그 앞에 "+"가 붙고 그뒤에 보통 프로그람이 실행될 때 나타나는것이 놓인다. read a 는 입력을 기다리도록 프로그람을 정지시키므로 프로그람의 그 위치에서 무엇이 수행되는가를 볼수 있다.

이 프로그람을 실행시키기 위하여서는 기본디스크장치를 포함하는 파일을 약간 변경하고 매행의 앞부분에 기록권그룹번호를 추가하여야 한다. 그리고 기본디스크를 가지고 있는 파일로부터 그 프로그람을 호출하고 디스크의 한개 행을 한번에 조작하여야 한다. 아래에서는 앞에서 본 쉘프로그람(vg.sh)의 이름을 포함하는 갱신된 파일을 보여 주고 있다. 그뒤에는 기록권그룹번호와 매 기록권그룹에 대한 다섯개 기본디스크이름들의 목록이 따른다.

```
#vg.sh  01   c9t0d0   c9t0d1   c9t0d2   c8t0d0    c8t0d1
#read a
#vg.sh  02   c7t0d0   c7t0d1   c7t0d2   c10t0d0   c10t0d1
#read a
#vg.sh  03   c9t0d3   c9t0d4   c9t0d5   c8t0d3    c8t0d4
#read a
#vg.sh  04   c7t0d3   c7t0d4   c7t0d5   c10t0d3   c10t0d4
#read a
#vg.sh  05   c9t0d6   c9t0d7   c9t1d0   c8t0d6    c8t0d7
#read a
```

```
#vg.sh    06    c7t0d6    c7t0d7    c7t1d0    c10t0d6    c10t0d7
#read  a
#vg.sh    07    c9t1d1    c9t1d2    c9t1d3    c8t1d1     c8t1d2
#read  a
#vg.sh    08    c7t1d1    c7t1d2    c7t1d3    c10t1d1    c10t1d2
#read  a
#vg.sh    09    c9t1d4    c9t1d5    c9t1d6    c8t1d4     c8t1d5
#read  a
#vg.sh    10    c7t1d4    c7t1d5    c7t1d6    c10t1d4    c10t1d5
```

이 파일의 행들사이에 있는 read a 는 다음 행이 실행되기전에 Return 을 입력할 때까지 중지하고 기다리게 한다. 이것은 여러 행을 실행시키고 행들이 실행되는 사이에 결과를 검사하려는 경우에 필요된다.

그 파일의 첫 행에 설명문을 붙이지 않고 파일이름 pri 를 입력할수 있으며 그러면 그 행은 vg.sh 를 호출하고 실행시킨다(그 파일들에 적당한 허락을 주어야 하며 vg.sh 와 pri 가 실행가능하다는것을 담보하여야 한다). 그러한 파일들은 한 행을 한번에 실행시키고 기록권그룹이 창조되자마자 검사될수 있다. 이 스크립트는 한 행을 한번에 실행하도록 작성되였으나 10 개 행인수를 실행하도록 쉽게 변경할수 있다.

교대디스크이름을 가지고서는 적게 작업하여야 한다. 물리적기록권은 이미 창조되였고 기록권그룹과 단순한 론리기록권은 이미 vg.sh 에 설치되였다. 매 디스크에 대한 교대이름을 포함하도록 기록권그룹을 확장한 vga.sh 라고 부르는 다른 스크립트를 창조한다. 이 스크립트를 아래에서 보여 주고 있다.

```
#!  /bin/ksh
set  -x              ;set  tracing  on

   vgnum=$1          ;first  item  on  each  line  is  the  volume  group  number
   shift             ;shift  to  get  to  first  disk

   for  i  in  $*    ;extend  vol  group  to  include  all  five  disks  on  line
          vgextend  /dev/vgu$vgnum  /dev/dsk/$i
   done
```

이 스크립트는 행에 나타나는 다섯개 디스크모두를 포함하도록 기록권그룹을 확장하는 과제만을 수행한다. pri 파일처럼 alt 파일은 vga.alt 스크립트를 호출하여 다음과 같은 결과를 얻는다.

```
#vga.sh   01    c8t8d0     c8t8d1     c8t8d2     c9t8d0     c9t8d1
#vga.sh   02   c10t8d0    c10t8d1    c10t8d2     c7t8d0     c7t8d1
```

```
#vga.sh  03   c8t8d3    c8t8d4    c8t8d5    c9t8d3   c9t8d4
#vga.sh  04  c10t8d3   c10t8d4   c10t8d5    c7t8d3   c7t8d4
#vga.sh  05   c8t8d6    c8t8d7    c8t9d0    c9t8d6   c9t8d7
#vga.sh  06  c10t8d6   c10t8d7   c10t9d0    c7t8d6   c7t8d7
#vga.sh  07   c8t9d1    c8t9d2    c8t9d3    c9t9d1   c9t9d2
#vga.sh  08  c10t9d1   c10t9d2   c10t9d3    c7t9d1   c7t9d2
#vga.sh  09   c8t9d4    c8t9d5    c8t9d6    c9t9d4   c9t9d5
#vga.sh  10  c10t9d4   c10t9d5   c10t9d6    c7t9d4   c7t9d5
```

스크립트를 어느 한 행에 대하여 실행시키려는 경우에는 그 행에 설명문을 달지 않아도 된다. 또한 10 개 행모두를 실행시킬수 있으나 매 행이 실행된 다음 무엇이 있는가를 검사하는것이 좋다. 여러 행을 실행시키고 결과를 검사하기 위하여서는 read a 를 추가하여 그것들사이에 휴식을 주어야 한다. 이 두 스크립트들은 입력을 자동화한다. 다른 론리기록권관리지령들은 물론 100 개의 디스크들에 대하여 지령들이 실현되여야 한다. 이것이 쉘프로그람작성을 위해 리상적으로 편리한 HP-UX 체계관리과제의 형태이다.

추가적인 4 개 디스크에 대하여 vgcreate 가 한것과 같은 작업 및 추가적인 9 개 디스크장치들에 대하여 vgextend 가 한것과 같은 작업을 추가적인 디스크들에 대하여 완성하기 위하여 실행되였어야 할 걸음들이 완성되였다. 실례들에서는 첫 디스크만을 포함하였으므로 초기걸음들을 볼수 있다. 1 차기록권 또는 교대기록권내에 RAID 준위를 설치하지 말아야 한다. 왜냐하면 설치가 XP256 에서 내부적으로 진행되기때문이다. 아래의 vgdisplay 용지는 다섯개의 1 차 및 교대디스크그룹을 가진 기록권그룹 vgu01 을 위하여 설치한 디스크들을 보여 준다.

```
# vgdisplay /dev/vgu0l -v

VG Name                  /dev/vgu01
VG Write Access          read/write
VG Status                available
Max LV                   255
Cur LV                   1
Open LV                  1
Max PV                   16
Cur PV                   5
Act PV                   5
Max PE per PV            1751
VGDA                     10
PE Size (Mbytes)         4
Total PE                 8755
Alloc PE                 8755
```

```
Free PE                     0
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0
   --- Logical volumes ---
   LV Name                  /dev/vgu01/lvol1
   LV Status                available/syncd
   LV Size（Mbytes）         35020
   Current LE               8755
   Allocated PE             8755
   Used PV                  5

   --- Physical volumes ---
   PV Name                  /dev/dsk/c9t0d0
   PV Name                  /dev/dsk/c8t8d0Alternate Link
   PV Status                available
   Total PE                 1751
   Free PE                          0

   PV Name                  /dev/dsk/c9t0d1
   PV Name                  /dev/dsk/c8t8d1Alternate Link
   PV Status                available
   Total PE                 1751
   Free PE                          0

   PV Name                  /dev/dsk/c9t0d2
   PV Name                  /dev/dsk/c8t8d2Alternate Link
   PV Status                available
   Total PE                 1751
   Free PE                          0

   PV Name                  /dev/dsk/c8t0d0
   PV Name                  /dev/dsk/c9t8d0Alternate Link
   PV Status                available
   Total PE                 1751
   Free PE                          0

   PV Name                  /dev/dsk/cBt0d1
   PV Name                  /dev/dsk/c9t8d1Alternate Link
```

PV Status available
Total PE 1751
Free PE 0

이 vgdisplay 에는 흥미 있는 문제들이 있다. 그것은 이미 앞에서 정의한바와 같이 동일한 디스크에로의 1 차 및 교대경로가 있는것이다. 실례로 기록권그룹의 첫 디스크는 c9t0d0 의 1 차경로이름과 c8t8d0 의 교대경로이름을 가진다. 다음 기록권그룹 vgu01 과 그 안의 론리적그룹 lvol1 은 8755PE 또는 물리적대역 전체로 이루어 진다.(기록권그룹의 크기는 PE x PE 또는 이 경우에 8755 x 4MB 이다.)

vgu01 우의 lvol1 이라고 부르는 한개 론리기록권을 검사할수도 있다. 이 론리기록권의 파라메터들을 아래 실례에서 보여 주는것처럼 lvdisplay 지령으로 검사할수 있다.

lvdisplay -v /dev/vgu01/l*

--- Logical volumes ---

LV Name	/dev/vgu01/lvol1
VG Name	/dev/vgu01
LV Permission	read/write
LV Status	available/syncd
Mirror copies	0
Consistency Recovery	MWC
Schedule	parallel
LV Size (Mbytes)	35020
Current LE	8755
Allocated PE	8755
Stripes	0
Stripe Size (Kbytes)	0
Bad block	on
Allocation	strict
10 Timeout (Seconds)	default

--- Distribution of logical volume ---

PV Name	LE on PV	PE on PV
/dev/dsk/c9t0d0	1751	1751
/dev/dsk/c9t0d1	1751	1751
/dev/dsk/c9t0d2	1751	1751
/dev/dsk/c8t0d0	1751	1751
/dev/dsk/c8t0d1	1751	1751

--- Logical extents ---

LE	PV1	PE1	Status 1
00000	/dev/dsk/c9t0d0	00000	current
00001	/dev/dsk/c9t0d0	00001	current
00002	/dev/dsk/c9t0d0	00002	current
00003	/dev/dsk/c9t0d0	00003	current
00004	/dev/dsk/c9t0d0	00004	current
00005	/dev/dsk/c9t0d0	00005	current
00006	/dev/dsk/c9t0d0	00006	current
00007	/dev/dsk/c9t0d0	00007	current
00008	/dev/dsk/c9t0d0	00008	current
00009	/dev/dsk/c9t0d0	00009	current
00010	/dev/dsk/c9t0d0	00010	current
00011	/dev/dsk/c9t0d0	00011	current
00012	/dev/dsk/c9t0d0	00012	current
00013	/dev/dsk/c9t0d0	00013	current
00014	/dev/dsk/c9t0d0	00014	current
00015	/dev/dsk/c9t0d0	00015	current
00016	/dev/dsk/c9t0d0	00016	current
00017	/dev/dsk/c9t0d0	00017	current
00018	/dev/dsk/c9t0d0	00018	current
00019	/dev/dsk/c9t0d0	00019	current
00020	/dev/dsk/c9t0d0	00020	current
00021	/dev/dsk/c9t0d0	00021	current
00022	/dev/dsk/c9t0d0	00022	current
00023	/dev/dsk/c9t0d0	00023	current
00024	/dev/dsk/c9t0d0	00024	current
00025	/dev/dsk/c9t0d0	00025	current
00026	/dev/dsk/c9t0d0	00026	current
00027	/dev/dsk/c9t0d0	00027	current
00028	/dev/dsk/c9t0d0	00028	current
00029	/dev/dsk/c9t0d0	00029	current
00030	/dev/dsk/c9t0d0	00030	current
00031	/dev/dsk/c9t0d0	00031	current
00032	/dev/dsk/c9t0d0	00032	current

⟆

08733	/dev/dsk/c8t0d1	01729	current

```
08734        /dev/dsk/c8t0d1   01730           current
08735        /dev/dsk/c8t0d1   01731           current
08736        /dev/dsk/c8t0d1   01732           current
08737        /dev/dsk/c8t0d1   01733           current
08738        /dev/dsk/cBt0d1   01734           current
08739        /dev/dsk/c8t0d1   01735           current
08740        /dev/dsk/c8t0d1   01736           current
08741        /dev/dsk/c8t0d1   01737           current
08742        /dev/dsk/c8t0d1   01738           current
08743        /dev/dsk/c8t0d1   01739           current
08744        /dev/dsk/c8t0d1   01740           current
08745        /dev/dsk/cBt0d1   01741           current
08746        /dev/dsk/c8t0d1   01742           current
08747        /dev/dsk/cBt0d1   01743           current
08748        /dev/dsk/c8t0d1   01744           current
08749        /dev/dsk/c8t0d1   01745           current
08750        /dev/dsk/c8t0d1   01746           current
08751        /dev/dsk/c8t0d1   01747           current
08752        /dev/dsk/c8t0d1   01748           current
08753        /dev/dsk/c8t0d1   01749           current
08754        /dev/dsk/c8t0d1   01750           current
```

이 출력은 세개의 점이 있는 곳에서 생략되였다. 여기서는 디스크의 시작부분과 마지막디스크의 끌부분만을 보여 주고 있다. lvdisplay 는 실지로 론리기록권을 이루고 있는 다섯개의 기본디스크들을 보여 준다. 마지막걸음은 vgu00 에 설치한 론리기록권에 파일체계를 마련하는것이다. 이 파제를 수행하는데는 SAM 이 편리하다. 이 절차를 제 8 장에서 SAM 을 리용하여 완성하였으므로 그것을 참조할수 있을것이다. 다음 항목에서는 C 쉘프로그람작성에 대한 간단한 개요를 고찰한다.

C 쉘프로그람작성

비록 쉘프로그람작성기술이 모든 쉘들에 적용되지만 일반적으로 C 쉘과 Korn 쉘사이에는 몇가지 차이점들이 존재한다. C 쉘을 리용하려면 여기에서 취급하는 C 쉘프로그람작성법의 기초들을 학습하는것이 필요하다. 매개 쉘프로그람작성기술을 간단히 취급하고 매 기술을 숙련하기 위한 기초적인 실례들을 보기로 한다.

아래의 모든 쉘프로그람들에서 "#"로 시작되는 행은 설명문이다. 보통 쉘프로그람의 제일 첫 행에는 그 프로그람의 목적을 서술하는 설명문을 기입한다. 아래의 모든 프로그람들에서 C 쉘은 #!/bin/csh 로 실행되는데 이것은 이 항목의 실례들에서 리용된 Solaris 체

계상에서 C쉘의 경로이다.

지령치환

앞에서 취급한 쉘변수들은 지령으로부터의 출력을 보관하는데 리용될수 있다. 또한 다른 지령들을 수행시킬 때에도 그 변수들을 리용할수 있다. 아래의 쉘프로그람은 date 지령을 실행하고 그 결과를 변수 d에 보관한다. 그러면 변수 d는 cdate 프로그람의 echo 지령내에서 리용된다.

```
# ! /bin/csh
# program "today" that provides the date
set d='date +%x'
echo "Today's date is $d"
```

cdate를 실행시키면 아래의 결과가 생겨난다.

```
martyp $ cdate
Today's date is 06/01/00
martyp $
```

우의 실례에서 "+%x"는 현재의 날자를 생성한다. 이런 형태의 지령치환은 여러개의 쉘스크립트들에서 리용된다.

사용자입력의 읽기

쉘프로그람에 사용자입력을 읽어 들이는 방법에는 두가지가 있다. 첫째로, 사용자에게 정보를 요구하는것이고 둘째로, 쉘프로그람에 인수들을 제공하는것이다.

사용자에게 정보를 요구하는것으로부터 시작하자. 변수에 기호, 단어, 문장을 읽어들일수 있다.

아래의 실례에서는 사용자에게 먼저 단어를 요구하고 그다음에 문장을 요구하는것을 보여 주고 있다.

```
#! /bin/csh
echo "Please enter your name:"
set name = $<
echo "hello, $name"
echo "Please enter your favorite quote:"
set quote = $<
echo "Your favorite quote is:"
```

```
echo $quote
```

이 프로그람의 실행실례는 다음과 같다.

```
martyp $ userinput
Please enter your name:
Marty
hello, Marty
Please enter your favorite quote:
Creating is the essence of life.
Your favorite quote is:
Creating is the essence of life.
martyp $
```

이 기술을 리용하여 쉘프로그람에서 사용자에게 정보를 요구할수 있다. 이 기술은 아래의 프로그람들에서 리용된다.

지령행인수를 입력할수도 있다. 쉘스크립트의 이름을 입력할 때 변수 $1-$9 들에 보관되는 인수들을 줄수 있다. 지령행의 첫 10 개 단어들은 특수한 변수 $1-$9 들을 리용하여 쉘프로그람에서 직접 얻을수 있다.

$0	지령이름
$1	첫번째 인수
$2	두번째 인수
⁞	
$9	아홉번째 인수

프로그람이 실행될 때 몇개의 지령행인수들을 얻을수 있는가 하는것이 확실하지 않을 때에는 다음의 두개 변수들을 리용할수 있다.

$#	지령행인수들의 개수
$*	모든 지령행인수들이 공백으로 분리된 목록(지령이름은 포함하지 않는다.)

변수 $*은 임의의 개수의 인수들을 가진 쉘스크립트지령행들을 처리하기 위하여 for 순환과 흔히 리용된다.

아래의 스크립트는 규정된 등록부($1)에로 변경하고 규정된 파일($3)에서 규정된 패턴을 탐색한다.

```
#! /bin/csh
```

```
# search
# Usage: search directory pattern file
echo " "
cd $1               # change to search dir and
grep -n "$2" $3     # search for $2 in $3
echo " "            # print line
endif
```

grep 는 페턴에 대한 파일을 탐색하고 그 패턴이 찾아 진 행을 인쇄한다. 앞에서 취급된 awk 는 규정된 마당을 행안에서 찾아 내는데 리용될수 있다. search 프로그람의 실례를 하나 고찰하자.

```
martyp $ search /home/martyp/shellprogs awk ifstat

12:# as one command so it can be easily piped to awk.
18:awk 'BEGIN { printf "%10s%10s%10s%10s%10s\n", "ipkts",
38:' # End of the awk program.

martyp $
```

이 실례에서는 파일 ifstat 에서 패턴 awk 를 찾기 위하여 search 를 등록부 /home/martyp/shellprogs 에서 실행시킨다. 이 탐색의 결과는 awk 가 나타나는 파일 ifstat 에 3개 행을 생성한다. 그것들의 행번호는 12, 18, 38 이다.

검사와 갈래

쉘프로그람이 수행될수 있는 **결심채택**(Decision-Making)의 종류는 여러가지 이다. if 는 결심을 내리고 알맞는 작용을 선택할수 있게 한다. 3 개 인수들이 제공되였다는것을 확인하도록 search 스크립트를 확장하자.

```
#!/bin/csh
# search
# Usage: search directory pattern files
if ($#argv != 3) then        # if < 3 args provided
   echo "Usage: search directory pattern files"
                             # then print Usage
else
   echo " "                            # else print line and
   cd $1                     # change to search dir
   grep -n "$2" $3                     # search for $2 in $3
```

```
    echo " "                        # print line
endif
```

이 프로그람을 search1 이라고 하고 search 프로그람에서와 같은 동일한 인수들을 리용하여 실행시키자. search1 을 실행시킬 때 인수들을 주지 않으면 usage 통보문을 현시한다. 아래의 실례에서는 search1 의 실행을 보여 주고 있다.

```
martyp $ search1
Usage: search directory pattern files
martyp $ search1 /home/martyp/shellprogs awk llsum
12:# drwxrwxrwx 2 gerry aec 24 Mar 21 18:25   awk_ex
15:# awk field numbers:
18:awk ' BEGIN { x=i=0; printf "%-16s%-10s%8s%8s\n", \
martyp $
```

search1 의 첫번째 실행에서는 인수들을 주지 않았다. 3 개보다 적은 인수들이 발견되였기때문에 프로그람은 3 개이하의 인수들을 주었는가 그리고 usage 통보문이 생성되였는가를 검사하였다. usage 통보문을 보고 프로그람을 어떻게 리용하겠는가 하는것이 명백해졌으며 그 프로그람의 두번째 실행에서는 요구되는 3 개 인수들을 주었다. 이 실례에서는 awk 패턴을 파일 llsum 에서 찾기 위하여 search1 을 등록부 /home/martyp/shellprogs 에서 실행하였다. 이 탐색의 결과는 awk 가 나타나는 파일 llsum 에 3 개 행을 생성한다. 그 행들의 번호는 12, 15, 18 이다. 흔히 리용되는 if 의 4 가지 형식은 다음과 같다.

```
1)      if (expression) command
2)      if (expression) then
                        command(s)
        endif
3)      if (expression) then
                        command(s)
                else
                        command(s)
        endif
4)      if (expression) then
                        command(s)
                [else if expression] then
                        command(s)]
                 ⁝
                [else
                        command(s)]
        endif
```

C쉘에서 옹근수비교에 리용될수 있는 연산자들은 다음과 같다.

>	보다 크다.
<	보다 작다.
>=	보다 크거나 같다.
<=	보다 작거나 같다.
==	같다.
!=	같지 않다.

순환

C쉘에는 **순환**(Loop)을 보장하는 여러가지 기능들이 있다.

1) foreach 순환은 항목들의 목록을 정하고 그 목록안의 매 항목에 대하여 순환안의 지령들을 한번씩 수행시킨다.
2) while 순환은 어느한 지령(test 지령과 같은)을 수행하고 그 지령이 성과적으로 수행되면 순환안의 지령들을 수행한다.

foreach 순환의 형식은 다음과 같다.

```
foreach  name(list)
        command(s)
end
```

아래의 실례는 foreach 순환을 리용하여 /etc/hosts 에 있는 체계들이 **국부호스트**(Local Host)에 련결되였는가를 검사한다.

```
#! /bin/csh
#Program name: csh-hostck

#This program will test connectivity to all other hosts in
#your network listed in your /etc/hosts file.

# It uses the awk command to get the names from the hosts file
#and the ping command to check connectivity.

#Note that we use /bin/echo because csh echo doesn't support
#escape chars like \t or \c which are used in the
#foreach loop.
```

```
#Any line in /etc/hosts that starts with a number represents
#a host entry. Anything else is a comment or a blank line.

#Find all lines in /etc/hosts that start with a number and
#print the second field (the hostname).

set hosts=`awk '/^[1-9]/ { print $2 }' /etc/hosts`
                # grave on outside, single quote on inside

    /bin/echo "Remote host connection status:"

foreach sys ($hosts)
    /bin/echo "$sys - \c"
            # send one 64 byte packet and look for
            # the"is alive" message in
            # the output that indicates success.
            # messages vary between UNIX variants.
    ping $sys        64 1 | grep "is alive" > /dev/null
    if ( $status = = 0 ) then
            echo "OK"
    else
            echo "DID NOT RESPOND"
    endif
end
```

awk 로 조사하는 행은 /etc/hosts 파일로부터 **원격호스트**(Remote Host)들의 이름을 얻는데 리용된다. foreach 순환은 목록안의 모든 항목들(이 경우에는 호스트들)을 선택하고 그것들의 상태를 검사한다. 체계상의 hosts 파일에는 3 개의 항목 즉 국부호스트, LAN 대면부 그리고 DNS 체계가 들어 있다. 아래에서 보여 준 실례의 프로그람을 실행시키면 그 세개 항목들에 대한 검사결과를 얻을수 있다.

```
martyp $ csh-hostck
Remote host connection status:
localhost - OK
sunsys - OK
dnssrv1 - OK
martyp $
```

hosts 파일안의 세개 항목들은 모두 ping 에 의해 평가되며 OK 라는 상태를 알려 준다.

ping의 경로와 지령 ping으로부터 얻은 결과와 같은 정보를 스크립트에 코드화하기 힘들면 그것들은 각이한 UNIX 변종들에서 변할수 있다는것을 알아야 한다. 쉘프로그람에 설명문을 붙이는 리유의 하나는 이와 같은 경우에 프로그람을 쉽게 변경할수 있기 위해서이다.

지령들을 여러번 반복하여 실행시키기 위하여 while 순환을 리용할수 있다. while 순환의 형식은 다음과 같다.

```
while (expression)
   command (s)
end
```

netcheck 라고 부르는 아래의 프로그람은 요구되는 간격으로 netstat 를 실행시키며 머리부를 한번 인쇄하고 le0의 상태를 9번 인쇄한다.

```
#! /bin/csh
# program to run netstat at every specified interval
# Usage: netcheck interval

set limit=9                  # set limit on number times
                             # to run netstat
echo " "
netstat -i | grep Name                 # print netstat line with head-
ings
set count=0
while ($count<$limit)        # if limit hasn't reached
                             # limit run netstat
          netstat -i | grep le0
          sleep $1           # sleep for interval
                             # specified on command line
          @ count++          # increment limit
end
echo "count has reached $limit, run netcheck again to see le0 status"
```

netcheck 프로그람의 실행실례는 다음과 같다.

martyp $ **netcheck 3**

Name	Mtu	Net/Dest	Address	Ipkts	Ierrs	Opkts	Oerrs	Collis	Queue
le0	1500	sunsys	sunsys	314374989	0	17252200	52135	7580906	

```
le0   1500   sunsys   sunsys   314375038   0  17252203   52135   7580906
le0   1500   sunsys   sunsys   314375114   0  17252206   52135   7580906
le0   1500   sunsys   sunsys   314375185   0  17252209   52135   7580906
le0   1500   sunsys   sunsys   314375257   0  17252212   52135   7580906
le0   1500   sunsys   sunsys   314375332   0  17252215   52135   7580906
le0   1500   sunsys   sunsys   314375444   0  17252218   52135   7580906
le0   1500   sunsys   sunsys   314375508   0  17252221   52135   7580906
le0   1500   sunsys   sunsys   314375588   0  17252224   52135   7580906
count has reached 9, run netcheck again to see le0 status
martyp $
```

netcheck 의 출력은 9 개의 netstat 출력을 생성한다. 체계상에서 이 프로그람을 리용할 때 그 정보를 LAN 대면부의 이름으로 변경하여야 한다.

이 프로그람은 다음과 같이 식의 값을 증가시킨다.

```
@ count ++
```

식이 진실이면 지령(들)이 수행된다. @count++는 다음과 같은 형식의 대입연산자이다.

```
@ variable_name operator expression
```

이 경우에 변수는 우선 "="에 의하여 대입되고 그다음에 자동적으로 증가(++)된다. 표 12-2 에 서술한바와 같이 변수에 적용될수 있는 연산은 여러가지이다.

표 12-2 대입연산자들

연 산	기호	실례(count=100)	결과
값을 보관한다	=	@count=100	100
자동증가	+ +	@count++	101
자동감소	- -	@count- -	99
값을 더한다	+ =	@count+=50	150
값을 던다	- =	@count- =50	50
값을 곱한다	* =	@count*=2	200
값으로 나눈다	/ =	@count/=2	50

다음으로 비교연산자, 산수연산자, 비트별연산자, 론리연산자들을 보기로 하자. 쉘스크립트를 보다 더 기교 있게 작성하려면 이 연산자들을 잘 리용하여야 한다.

파일을 리용하는 쉘스크립트를 작성할 때 리용될수 있는 파일관련의 검사조건들이 있다. operator -filename 형식을 리용할 때 표 12-3 에 있는 검사들을 리용할수 있다.

filetest 라고 부르는 아래의 프로그람은 파일 .profile 을 검사하는데 이 연산자들을 리용한다. .profile 은 실행불가능하기때문에 filetest 는 그것을 허위로 간주한다.

.profile은 다음과 같다.

```
martyp $ ls -al .profile
-rw-r--r-- 1 martyp staff 594 May 21 09:29 .. / .profile
martyp $
```

표 12-3 파일이름검사를 위한 연산자들

연산자	의 미
r	읽기접근
w	쓰기접근
x	실행접근
o	소유기록권
z	령길이
f	등록부가 아니라 파일
d	파일이 아니라 등록부

쉘스크립트 filetest는 다음과 같다.

```
#! /bin/csh
# Program to test file $1

if (-e $1) then
        echo "$1 exists"
        else
        echo "$1 does not exist"
endif

if (-z $1) then
        echo "$1 is zero length"
        else
        echo "$1 is not zero length"
endif

if (-f $1) then
        echo "$1 is a file"
        else
        echo "$1 is not a file"
endif
```

```
if (-d $1) then
        echo "$1 is a directory"
        else
        echo "$1 is not a directory"
endif
if (-o $1) then
        echo "you own $1"
        else
        echo "you don't own $1"
endif

if (-r $1) then
        echo "$1 is readable"
        else
        echo "$1 is not readable"
endif

if (-w $1) then
        echo "$1 is writable"
        else
        echo "$1 is not writable"
endif

if (-x $1) then
        echo "$1 is executable"
        else
        echo "$1 is not executable"
endif
```

.profile을 입력으로 리용하는 filetest의 출력은 다음과 같다.

martyp $ **filetest /home/martyp/.profile**

```
/home/martyp/.profile exists
/home/martyp/.profile is not zero length
/home/martyp/.profile is a file
/home/martyp/.profile is not a directory
you own /home/martyp/.profile
/home/martyp/.profile is readable
/home/martyp/.profile is writable
/home/martyp/.profile is not executable
martyp $
```

.profile 상에서 filetest 가 실행된 결과는 요구하는 파일검사결과를 준다.

switch 에 의한 판단

쉘프로그람안에서 결심을 채택하는데 switch 를 리용할수 있다. 또한 지령행인수를 검사하거나 쉘프로그람에 입력을 진행할 때 switch 를 리용할수 있다. 실례로 쉘스크립트에 차림표를 창조하고 그 프로그람이 실행될 때 사용자가 어느 인수를 선택하였는가를 결정하려는 경우에 switch 를 리용할수 있다.

switch 의 문장형식은 다음과 같다.

```
switch（pattern_to_match）
 case  patternl
          commands
          breaksw
 case  pattern2
          commands
          breaksw
 case  pattern  3
          commands
          breaksw
  default
          commands
          breaksw
endsw
```

pattern_to_match 는 검사하려는 사용자입력인데 그것이 pattern1 과 같으면 pattern1 아래에 있는 지령들이 실행된다. pattern_to_match 와 pattern2 가 동일하면 pattern2 아래의 지령들이 실행된다. pattern_to_match 와 맞는 패턴이 case 명령문에 없으면 default 아래의 지령들이 수행된다. 아래의 프로그람은 차림표상에서 두 스크립트들중 하나를 골라 낼수 있게 한다. 우에서 이미 작성된 두개의 쉘프로그람에 요구하는 프로그람을 더 첨가하여 스크립트를 확장할수 있다. 아래의 실례에서는 switch 를 리용하고 있다.

```
#!  /bin/csh
# Program  pickscript  to  run  some  of
# the  C  shell  scripts  we've  created
# Usage:  pickscript
echo  "----------------------------------------------------------------------------"
echo  "              Sys  Admin  Menu                                            "
echo  "----------------------------------------------------------------------------"
echo  "       "
```

```
echo "      1      netcheck  for  network  interface                              "
echo "      "
echo "      2      hostck  to  check  connection                                  "
echo "             to  hosts  in  /etc/hosts                                      "
echo "                                                                            "
echo "--------------------------------------------------------------------------------"
echo "      "
echo " Please  enter  your  selection  ->  \c"
set  pick  =  $<       #  read  input  which  is  number  of  script
echo  "  "
switch  ($pick)       #  and  assign  to  variable  pick
          case  1           #  if  1  was  selected  execute  this
          $HOME/cshscripts/netcheck  5
          breaksw
          case  2           #  if  2  was  selected,  execute  this
          $HOME/cshscripts/hostck
          breaksw
          default
          echo  "Please  select  1  or  2  next  time"
          breaksw
endsw
```

이 프로그람은 두 스크립트들중에서 실행하려는것을 선택할수 있게 한다. 이 프로그람이 실행된 실례는 다음과 같다.

```
martyp  $  pickscript
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
          Sys   Admin   Menu
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  1   netcheck  for  network  interface
  2   hostck  to  check  connection
        to  hosts  in  /etc/hosts
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

  Please  enter  your  selection
1
Name      Mtu    Net/Dest   Address    Ipkts      Ierrs    Opkts      Oerrs     Collis     Queue
le0       1500   sunsys     sunsys     314996747  0        17261251   52135     7580952
le0       1500   sunsys     sunsys     314996862  0        17261256   52135     7580952
```

```
le0     1500    sunsys    sunsys     314997189   0      17261266    52135    7580952
le0     1500    sunsys    sunsys     314997319   0      17261269    52135    7580952
le0     1500    sunsys    sunsys     314997420   0      17261272    52135    7580952
le0     1500    sunsys    sunsys     314997630   0      17261275    52135    7580952
le0     1500    sunsys    sunsys     314997774   0      17261278    52135    7580952
le0     1500    sunsys    sunsys     314997904   0      17261281    52135    7580952
le0     1500    sunsys    sunsys     314998020   0      17261284    52135    7580952
count has reached 9, run netcheck again to see lan0 status
martyp $
```

pickscript 가 실행될 때 선택항목 1 을 선택하였다. 이 프로그람은 쉘프로그람들을 실행시키기 위한 기초로 리용될수 있다.

C 쉘프로그람의 오유수정

C 쉘프로그람작성을 시작할 때 단순한 문법오유들을 범할수 있다. csh 에 선택항목 -n 을 주면 C 쉘은 프로그람을 실행시키지 않고 문법을 검사한다. 선택항목 –v 를 주면 많은 정보들이 출력된다. 그러므로 -v 로 시작하는것이 지나치게 많은 결과가 생겨 나면 그것을 생략한다.

아래의 실례는 이미 앞에서 고찰한 search1 프로그람인데 3 개 인수가 주어 졌는가 하는것을 검사한다. $#argv 가 3 과 같은가를 검사할 때 오른쪽괄호를 제거하였다. 아래에 그 프로그람과 문법검사결과를 보여 준다.

```
martyp $ cat search1
#!/bin/csh
# search
# usage: search directory pattern files
if ($#argv !3 3 then                    # if < 3 args provided
  echo "Usage: search directory pattern files"
# then print Usage
else
  echo " "                              # else print line and
  cd $1                                          # change to search dir
  grep -n "$2" $3                       # search for $2 in $3
  echo " "                              # print line
endif
```

```
martyp $ csh -nv search1
if ( $#argv != 3 then
Too many ('s
martyp $
```

csh **-nv** 는 문법검사를 하면서 불필요한것을 출력하였다. 우선 질문행이 인쇄되였으며 그다음에 그 행에서 무엇이 틀렸는가를 말해 주는 오유통보문이 인쇄되였다. 이 경우에 오른쪽 괄호가 없다는것이 명백하다. 오유를 퇴치한후에 **-x** 로 프로그람을 실행시키면 모든 지령들은 실행되기전에 즉시 현시된다.

아래의 실례에서는 search 프로그람의 실행프로쎄스를 보여 주고 있다.

```
martyp $ csh -xv searchl shellprogs grep csh_hostck

if （ $#argv ! = 3 ） then
if （ 3 ! = 3 ） then

echo " "
echo

cd $l
cd /home/martyp/shellprogs
grep -n " $2 "  $3
grep -n grep csh_hostck
25:  /usr/sbin /ping $sys 64 1 | grep "is alive" > /dev/
null
echo " "
echo

endif
endif
martyp $
```

25 로 시작되는 행은 grep 가 들어 있는 파일 csh_hostck 에 있는 행이다. 즉 선택항목 -xv 가 없이 프로그람이 실행되는 경우에 얻게 되는 출력이다.

새로운 쉘프로그람에서는 문법검사(-n)를 수행하는것이 좋다. 그리고 프로그람이 실행될 때 기대하지 않았던 결과가 얻어 질 때에는 선택항목 -x 로 모든 지령들을 현시해야 한다. 오유수정은 쉘프로그람의 작성과 수정을 돕는 좋은 수단이다.

얼마나 오래 걸리는가

time 지령을 리용하면 쉘프로그람이 실행되는데 걸린 시간을 알수 있다. time 출력은 UNIX 변종에 따라 각이하다. 그림 12−5 에서는 csh 에서 time 의 출력을 보여 주고 있다.

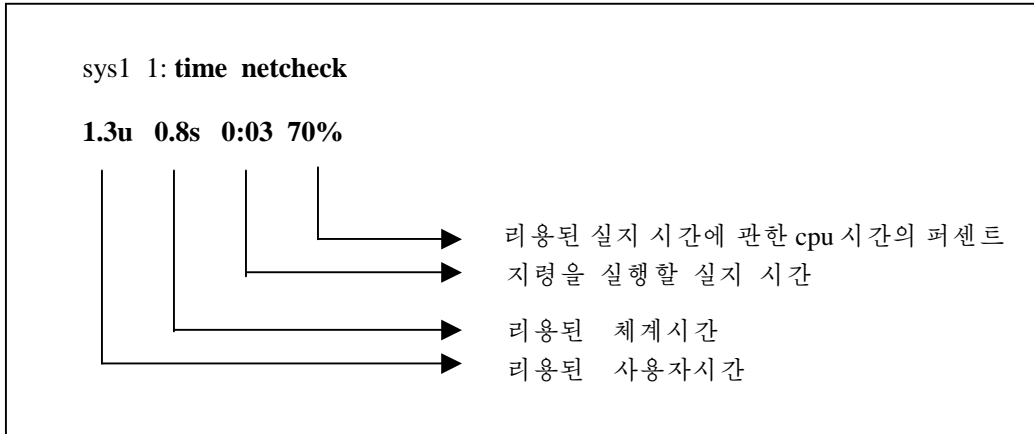



그림 12-5. time 실례(UNIX 변종들에서 차이난다.)

일부 스크립트들은 체계자원을 소비할수 있기때문에 C 쉘의 **일감**(job)조종능력을 고려하여야 한다. 리용될수 있는 가장 간단한 일감조종은 스크립트의 우선권이 낮아 지도록 스크립트들을 배경에서 실행시키는것이다. 스크립트의 이름뒤에 &를 붙이면 그 스크립트는 배경에서 실행된다. 여러개의 스크립트들이 배경에서 실행되는 경우에 지령 jobs로 그것들의 상태를 얻을수 있다.

지령소개

이 장에서 리용된 지령들에 대한 지령들은 다른 장들에서 보여 준다.

ksh – 10장을 참고하시오.
grep, **awk**, **sed** 그리고 다른 도구들 – 6장을 참고하시오.

제１３장. 체계관리에 대한 개괄

체계관리

앞장에서는 리용가능한 디스크공간을 보면서 프로쎄스들을 렬거하거나 제거하는것과 같은 몇가지 체계관리에 대하여 취급하였다. 이 장에서는 많은 **비강압적인**(Unintrusive) 체계관리지령들을 고찰한다. 비강압적인 지령이라는것은 체계의 설치를 변경시키지 않는 지령들과 사용자가 입력하여 실행시키는 지령들을 말한다.

체계관리자는 체계의 설치와 조작을 책임진다. 그러나 사용자들도 역시 여러 면에서 체계설치에 대하여 흥미를 가진다. 여기서는 체계설치와 관련하여 가장 흔히 리용되는 지령들과 체계가 어떻게 실행되는가를 취급하려고 한다. 사용자들은 자기의 체계가 작업하는 방식에 대하여 그리고 체계관리와 관련된 지령들, 실행중에 있는 프로쎄스들을 렬거하며 그것들을 제거하는데 직접 관련되는 지령들에 흥미를 가진다.

이 장의 마지막부분에 이 장에서 리용되는 많은 지령들에 대한 안내페지를 주었다. 체계관리와 관련된 많은 내용들은 사용자참고서에서 다양하게 취급된다. 안내페지에는 추가적으로 상세한것들이 서술되여 있기는 하지만 독자들을 아직 체계관리전문가수준으로 되게 하지는 못한다.

이 장의 실례들은 Solaris, HP-UX, AIX, Linux 체계들에서 수행된것들이다. 그러나 지령들과 파일들은 모든 체계들에서 류사하기때문에 이 장의 내용들을 다른 UNIX 변종과 련관시킬수 있다.

대부분의 UNIX 변종들은 체계관리도구를 갖추고 있다. 이 도구는 사용자와 그룹, 디스크와 파일체계, 망, 인쇄기 등을 관리하는것과 같은 공통적인 체계관리과제들을 **차림표구동형**(Menu-Driven)으로 수행시킨다. 널리 리용되는 표준적인 관리도구들은 없으므로 매 UNIX 변종에 따라 관리도구들은 서로 다르다. 이 도구들을 실행시키자면 **상급사용자접근**(Super User Access)이 요구된다.

일부 도구들은 상급사용자가 아닌 사용자들이 상급사용자가 정의한 부분과제들을 수행하도록 한다. UNIX 변종들은 자기의 고유한 체계관리도구들을 가지고 있으며 대체로 체계관리자에 의해 리용되기때문에 여기서는 체계관리도구들에 대하여 취급하지 않는다.

여기서는 체계관리자에 의하여 리용되는 파일들과 흔히 리용되는 비강압적인 일부 지령들을 취급하려고 한다. 아래의 일부 항목들은 다른 장들에서도 찾아 볼수 있다.

ps로 프로쎄스들을 검사

《지금 체계는 무엇을 하고 있는가?》에 대한 대답을 찾으려면 ps -ef를 리용한다. 이 지령은 체계에서 실행되고 있는 매 프로쎄스들에 대한 정보를 제공해 준다. 실례로 NFS가 실행되고 있는가를 알려면 간단히 ps -ef를 입력하고 NFS **데몬**(Daemon)을 찾으

면 된다. ps 는 체계에서 실행되는 모든 프로쎄스들을 알려 주기는 하지만 사용하고 있는 체계자원의 수준에 대하여서는 잘 알려 주지 못한다. ps 는 가장 자주 리용되는 체계관리지령이다. ps 에는 많은 선택항목들이 있다. 보통 e 와 f 를 리용하는데 e 는 실행되는 매 프로쎄스에 대한 정보를 제공하며 f 는 이 정보를 모두 렬거한다. ps 출력은 모든 체계들에서 거의 같다.

아래에서는 각각 Solaris, AIX, HP-UX 체계들에 대한 세개의 실례들을 보여 주고 있다.

Solaris 의 실례:

```
martyp $ ps -ef
    UID    PID  PPID   C     STIME   TTY  TIME    CMD
   root      0     0   0  Feb  18      ?    0:01  sched
   root      1     0   0  Feb  18      ?    1:30  /etc/init-
   root      2     0   0  Feb  18      ?    0:02  pageout
   root      3     0   1  Feb  18      ?  613:44  fsflush
   root   3065  3059   0  Feb  22      ?    5:10  /usr/dt/bin/sdtperfmeter -f -H -r
   root     88     1   0  Feb  18      ?    0:01  /usr/sbin/in.routed -q
   root    478     1   0  Feb  18      ?    0:00  /usr/lib/saf/sac -t 300
   root     94     1   0  Feb  18      ?    2:50  /usr/sbin/rpcbind
   root    150     1   0  Feb  18      ?    6:03  /usr/sbin/syslogd
   root     96     1   0  Feb  18      ?    0:00  /usr/sbin/keyserv
   root    144     1   0  Feb  18      ?   50:37  /usr/lib/autofs/automountd
   root   1010     1   0  Apr  12      ?    0:00  /opt/perf/bin/midaemon
   root    106     1   0  Feb  18      ?    0:02  /usr/lib/netsvc/yp/ypbind -broadt
   root    156     1   0  Feb  18      ?    0:03  /usr/sbin/cron
   root    176     1   0  Feb  18      ?    0:00  /usr/lib/lpsched
   root    129     1   0  Feb  18      ?    0:00  /usr/lib/nfs/lockd
 daemon    130     1   0  Feb  18      ?    0:01  /usr/lib/nfs/statd
   root  14798     1   0  Mar  09      ?   31:10  /usr/sbin/nscd
   root    133     1   0  Feb  18      ?    0:10  /usr/sbin/inetd -s
   root    197     1   0  Feb  18      ?    0:00  /usr/lib/power/powerd
   root    196     1   0  Feb  18      ?    0:35  /etc/opt/licenses/lmgrd.ste -c /d
   root    213     1   0  Feb  18      ? 4903:09  /usr/sbin/vold
   root    199   196   0  Feb  18      ?    0:03  suntechd -T 4 -c /etc/optd
   root    219     1   0  Feb  18      ?    0:08  /usr/lib/sendmail  -bd  -ql5m
   root    209     1   0  Feb  18      ?    0:05  /usr/lib/utmpd
   root   2935   266   0  Feb  22      ?   48:08  /usr/openwin/bin/Xsun  :0  -nobanna
   root  16795 16763   1  07:51:34 pts/4    0:00  ps -ef
   root   2963  2954   0  Feb  22      ?    0:17  /usr/openwin/bin/fbconsole
```

```
     root    479      1    0   Feb  18console   0:00  /usr/lib/saf/ttymon -g -h -p sunc
     root  10976      1    0   Jun  01     ?    0:00  /opt/perf/bin/ttd
     root   7468      1    0   Feb  24     ?    0:13  /opt/perf/bin/pvalarmd
     root    266      1    0   Feb  18     ?    0:01  /usr/dt/bin/dtlogin  daemon
   martyp  16763 16761    0  07:46:46  pts/4    0:01  -ksh
     root  10995      1    0   Jun  01     ?    0:01  /opt/perf/bin/perflbd
     root    484    478    0   Feb  18     ?    0:00  /usr/lib/saf/ttymon
     root    458      1    0   Feb  18     ?   20:06  /usr/lib/snmp/snmpdx -y -c /etc/f
     root  16792   3059    0  07:50:37     ?    0:00  /usr/dt/bin/dtscreen -mode blank
     root    471      1    0   Feb  18     ?    0:07  /usr/lib/dmi/dmispd
     root    474      1    0   Feb  18     ?    0:00  /usr/lib/dmi/snmpXdmid -s
     root    485    458    0   Feb  18     ?  739:44  mibiisa -r -p 32874
     root   2954   2936    0   Feb  22     ?    0:01  /bin/ksh /usr/dt/bin/Xsession
     root   2936    266    0   Feb  22     ?    0:00  /usr/dt/bin/dtlogin -daemon
     root   3061   3059    0   Feb  22     ?    1:32  dtwm
     root   3058      1    0   Feb  22  pts/2    0:01  /usr/dt/bin/ttsession
     root    712    133    0   Feb  18     ?    0:01  rpc.ttdbserverd
     root  11001 11000    0                     0:01  <defunct>
     root   2938      1    0   Feb  22     ?    0:00  /usr/openwin/bin/fbconsole -d :0
     root   2999   2954    0   Feb  22  pts/2    0:16  /usr/dt/bin/sdt-shell –c    unt
     root   3059   3002    0   Feb  22  pts/2  283:35  /usr/dt/bin/dtsession
     root   3063   3059    0   Feb  22     ?    0:03  /usr/dt/bin/dthelpview -helpVolur
     root   3099   3062    0   Feb  22     ?    0:13  /usr/dt/bin/dtfile -geometry +700
     root  11000 10995    0   Jun  01     ?    0:02  /opt/perf/bin/agdbserver -t alar/
     root   3002   2999    0   Feb  22  pts/2    0:01  -ksh –c    unset DT;    DISPLg
     root    730    133    0   Feb  18     ?    1:37  rpc.rstatd
     root   3062   3059    0   Feb  22     ?    2:17  /usr/dt/bin/dtfile -geometry +700
     root   3067      1    0   Feb  22     ?    0:00  /bin/ksh /usr/dt/bin/sdtvolcheckm
     root   3000      1    0   Feb  22     ?    0:00  /usr/dt/bin/dsdm
     root   3078   3067    0   Feb  22     ?    0:00  /bin/cat /tmp/.removable/notifyo
     root  10984      1    0   Jun  01     ?   12:42  /opt/perf/dce/bin/dced -b
     root  16761    133    0  07:46:45     ?    0:00  in.telnetd
martyp $
```

AIX 의 실례:

```
martyp $ ps -ef
UID     PID    PPID   C    STIME      TTY      TIME   CMD
root      1       0   0   Feb  24       -      5:07   /etc/init
```

```
root   2208  15520  0  Feb  24    -     8:21  dtwm
root   2664      1  0  Feb  24    -     0:00  /usr/dt/bin/dtlogin -daemon
root   2882      1  0  Feb  24    -   158:41  /usr/sbin/syncd 60
root   3376   2664  5  Feb  24    -  3598:41  /usr/lpp/Xii/bin/X -D /usr/lib/
root   3624   2664  0  Feb  24    -     0:00  dtlogin <:0>     -daemon
root   3950      1  6  Feb  24    -  5550:30  /usr/lpp/perf/bin/llbd
root   4144      1  0  Feb  24    -     0:00  /usr/lpp/perf/bin/midaemon
root   4490      1  0  Feb  24    -     0:48  /usr/lpp/perf/bin/perflbd
root   4906      1  0  Feb  24    -     0:00  /usr/lib/errdemon
root   5172      1  0  Feb  24    -     0:00  /usr/sbin/srcmstr
root   5724   5172  0  Feb  24    -     9:54  /usr/sbin/syslogd
root   6242   5172  0  Feb  24    -     0:00  /usr/sbin/biod 6
root   6450   5172  0  Feb  24    -     0:02  sendmail: accepting connections
root   6710   5172  0  Feb  24    -     7:34  /usr/sbin/portmap
root   6966   5172  0  Feb  24    -     0:23  /usr/sbin/inetd
root   7224   5172  0  Feb  24    -     1:09  /usr/sbin/timed -S
root   7482   5172  0  Feb  24    -    11:55  /usr/sbin/snmpd
root   8000      1  0  Feb  24    -     9:17  ovspmd
root   8516   8782  0  Feb  24    -     0:00  netfmt -CF
root   8782      1  0  Feb  24    -     0:00  /usr/OV/bin/ntl reader 0 1 1 1
root   9036   8000  0  Feb  24    -    10:09  ovwdb -0 -n5000
root   9288   8000  0  Feb  24    -     0:44  pmd -Au -At -Mu -Mt -m
root   9546   8000  0  Feb  24    -    20:05  trapgend -f
root   9804   8000  0  Feb  24    -     0:28  trapd
root  10062   8000  0  Feb  24    -     0:47  orsd
root  10320   8000  0  Feb  24    -     0:33  ovesmd
root  10578   8000  0  Feb  24    -     0:30  ovelmd
root  10836   8000  0  Feb  24    -    13:12  ovtopmd -0
root  11094   8000  0  Feb  24    -    17:50  netmon -P
root  11352   8000  0  Feb  24    -     0:02  snmpCollect
root  11954      1  0  Feb  24    -     1:22  /usr/sbin/cron
root  12140   5172  0  Feb  24    -     0:01  /usr/lib/netsvc/yp/ypbind
root  12394   5172  0  Feb  24    -     1:39  /usr/sbin/rpc.mountd
root  12652   5172  0  Feb  24    -     0:29  /usr/sbin/nfsd 8
root  12908   5172  0  Feb  24    -     0:00  /usr/sbin/rpc.statd
root  13166   5172  0  Feb  24    -     0:29  /usr/sbin/rpc.lockd
root  13428      1  0  Feb  24    -     0:00  /usr/sbin/uprintfd
root  14190   5172  0  Feb  24    -    72:59  /usr/sbin/automountd
root  14452   5172  0  Feb  24    -     0:17  /usr/sbin/qdaemon
```

```
    root   14714   5172   0  Feb  24      -     0:00  /usr/sbin/writesrv
    root   14992      1   0  Feb  24      -   252:26  /usr/lpp/perf/bin/scopeux
    root   15520   3624   1  Feb  24      -    15:29  /usr/dt/bin/dtsession
    root   15742      1   0  Feb  24      -     0:00  /usr/lpp/diagnostics/bin/diagd
    root   15998      1   0  Feb  24   1ft0     0:00  /usr/sbin/getty /dev/console
    root   16304  18892   0  Feb  24  pts/0     0:00  /bin/ksh
    root   16774      1   0  Feb  24      -     0:00  /usr/lpp/perf/bin/ttd
    root   17092   4490   0  Feb  24      -    68:54  /usr/lpp/perf/bin/rep_server -t
    root   17370  19186   3                     0:00  <defunct>
    root   17630  15520   0  Mar  25      -     0:00  /usr/dt/bin/dtexec -open 0 -ttp
    root   17898  15520   0  Mar  20      -     0:00  /usr/dt/bin/dtexec -open 0 -ttp
    root   18118  19888   0  Feb  24  pts/1     0:00  /bin/ksh
    root   18366   6966   0  Feb  24      -     0:00  rpc.ttdbserver 100083 1
    root   18446  15520   0  Mar  15      -     0:00  /usr/dt/bin/dtexec -open 0 -ttp
    root   18892  15520   0  Feb  24      -     3:46  /usr/dt/bin/dtterm
    root   19186  16304   0  Feb  24  pts/0     0:01  /usr/1pp/X11/bin/msmit
    root   19450      1   0  Feb  24      -    26:53  /usr/dt/bin/ttsession -s
    root   19684   2208   0  Feb  24      -     0:00  /usr/dt/bin/dtexec -open 0 -ttp
    root   19888  19684   0  Feb  24      -     0:00  /usr/dt/bin/dtterm
    root   20104  15520   0  Feb  27      -     0:00  /usr/dt/bin/dtexec -open 0 -ttp
    root   20248  20104   0  Feb  27      -     0:03  /usr/dt/bin/dtscreen
    root   20542  29708   0  May  14      -     0:03  /usr/dt/bin/dtscreen
    root   20912  26306   0  Apr  05      -     0:03  /usr/dt/bin/dtscreen
    root   33558      1   0  May  18      -     3:28  /usr/atria/etc/lockmgr -a /var/
    root   33834   6966   3  07:55:49     -     0:00  telnetd
    root   34072      1   0  May  18      -     0:00  /usr/atria/etc/albd-server
  martyp   36296  36608  13  07:56:07  pts/2    0:00  ps -ef
  martyp   36608  33834   1  07:55:50  pts/2    0:00  -ksh
    root   37220  15520   0  May  28      -     0:00  /usr/dt/bin/dtexec -open  0  -ttp
martyp  $
```

HP-UX 의 실례 (부분적임):

```
martyp  $  ps  -ef
UID      PID    PPID    C   STIME   TTY   TIME     COMMAND
root       0       0    0   Mar 9     ?   107:28   swapper
root       1       0    0   Mar 9     ?     2:27   init
root       2       0    0   Mar 9     ?    14:13   vhand
root       3       0    0   Mar 9     ?   114:55   statdaemon
```

```
root       4   0   0  Mar 9     ?    5:57  unhashdaemon
root       7   0   0  Mar 9     ? 154:33  ttisr
root      70   0   0  Mar 9     ?    0:01  lvmkd
root      71   0   0  Mar 9     ?    0:01  lvmkd
root      72   0   0  Mar 9     ?    0:01  lvmkd
root      13   0   0  Mar 9     ?    9:54  vx_sched_thread
root      14   0   0  Mar 9     ?    1:54  vx_iflush_thread
root      15   0   0  Mar 9     ?    2:06  vx_ifree_thread
root      16   0   0  Mar 9     ?    2:27  vx_inactive_cache_thread
root      17   0   0  Mar 9     ?    0:40  vx_delxwri_thread
root      18   0   0  Mar 9     ?    0:33  vx_logflush_thread
root      19   0   0  Mar 9     ?    0:07  vx_attrsync_thread

                        ⋮

root      69   0   0  Mar 9     ?    0:09  vx_inactive_thread
root      73   0   0  Mar 9     ?    0:01  lvmkd
root      74   0  19  Mar 9     ? 3605:29  netisr
root      75   0   0  Mar 9     ?    0:18  netisr
root      76   0   0  Mar 9     ?    0:17  netisr
root      77   0   0  Mar 9     ?    0:14  netisr
root      78   0   0  Mar 9     ?    0:48  nvsisr
root      79   0   0  Mar 9     ?    0:00  supsched
root      80   0   0  Mar 9     ?    0:00  smpsched
root      81   0   0  Mar 9     ?    0:00  smpsched
root      82   0   0  Mar 9     ?    0:00  sblksched
root      83   0   0  Mar 9     ?    0:00  sblksched
root      84   0   0  Mar 9     ?    0:00  strmem
root      85   0   0  Mar 9     ?    0:00  strweld
root    3730   1   0  16:39:22  console 0:00  /usr/sbin/getty console console
root     404   1   0  Mar 9     ?    3:57  /usr/sbin/swagentd
oracle   919   1   0  15:23:23  ?    0:00  oraclegprd（LOCAL=NO）
root     289   1   2  Mar 9     ?   78:34  /usr/sbin/syncer
root     426   1   0  Mar 9     ?    0:10  /usr/sbin/syslogd -D
root     576   1   0  Mar 9     ?    0:00  /usr/sbin/portmap
root     429   1   0  Mar 9     ?    0:00  /usr/sbin/ptydaemon
root     590   1   0  Mar 9     ?    0:00  /usr/sbin/biod 4
root     442   1   0  Mar 9     ?    0:00  /usr/lbin/nktl daemon 000001-2
oracle  8145   1   0  12:02:48  ?    0:00  oraclegprd（LOCAL=NO）
```

```
root     591     1    0    Mar 9     ?   0:00   /usr/sbin/biod 4
root     589     1    0    Mar 9     ?   0:00   /usr/sbin/biod 4
root     592     1    0    Mar 9     ?   0:00   /usr/sbin/biod 4
root     604     1    0    Mar 9     ?   0:00   /usr/sbin/rpc.lockd
root     598     1    0    Mar 9     ?   0:00   /usr/sbin/rpc.statd
root     610     1    0    Mar 9     ?   0:16   /usr/sbin/automount -f /etc/auto_master
root     638     1    0    Mar 9     ?   0:06   sendmail: accepting connections
root     618     1    0    Mar 9     ?   0:02   /usr/sbin/inetd
root     645     1    0    Mar 9     ?   5:01   /usr/sbin/snmpdm
root     661     1    0    Mar 9     ?  11:28   /usr/sbin/fddisubagtd
root     711     1    0    Mar 9     ?  30:59   /opt/dce/sbin/rpcd
root     720     1    0    Mar 9     ?   0:00   /usr/sbin/vtdaemon
root     867   777    1    Mar 9     ?   0:00   <defunct>
lp       733     1    0    Mar 9     ?   0:00   /usr/sbin/lpsched
root     777     1    0    Mar 9     ?   8:55   DIAGMON
root     742     1    0    Mar 9     ?   0:15   /usr/sbin/cron
oracle  7880     1    0  11:43:47    ?   0:00   oraclegprd (LOCAL=NO)
root     842     1    0    Mar 9     ?   0:00   /usr/vue/bin/vuelogin
oracle  5625     1    0  07:00:14    ?   0:01   ora_smon_gprd
root     781     1    0    Mar 9     ?   0:00   /usr/sbin/envd
root     833   777    0    Mar 9     ?   0:00   DEMLOG DEMLOG;DEMLOG;0; 0;
root     813     1    0    Mar 9     ?   0:00   /usr/sbin/nfsd 4
root     807     1    0    Mar 9     ?   0:00   /usr/sbin/rpc.mountd
root     815   813    0    Mar 9     ?   0:00   /usr/sbin/nfsd 4
root     817   813    0    Mar 9     ?   0:00   /usr/sbin/nfsd 4
root     835   777    0    Mar 9     ?   0:13   PSMONPSMON;PSMON;0;0;
```

머리부에 대한 간단한 서술을 하면 다음과 같다.

UID　　프로세스소유자의 사용자 ID
PID　　프로세스 ID(이 번호를 프로세스를 제거하는데 리용할수 있다.)
PPID　　선조프로세스의 프로세스 ID
C　　자원분배를 위한 프로세스리용
STIME　　프로세스의 시작시간
TTY　　프로세스를 조종하는 말단
TIME　　프로세스에 대한 루적실행시간
CIMMAND　　지령이름과 인수들

ps 는 체계에서 실행되고 있는 프로쎄스들을 륜곽적으로 보여 준다. 더 상세한 정보를 보려면 선택항목 "l"을 주면 된다. 그러면 아래의 실례에서 보여 주는바와 같이 많은 추가적인 정보를 제시하여 준다.

martyp $ **ps -efl**

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	STIME	TTY	D
19	T	root	0	0	0	0	SY	f026f7f0	0		Feb 18	?	d
8	S	root	1	0	0	41	20	f5b90808	175	f5b90a30	Feb 18	?	-
19	S	root	2	0	0	0	SY	f5b90108	0	f0283fd0	Feb 18	?	t
19	S	root	3	0	0	0	SY	f5b8fa08	0	f0287a44	Feb 18	?	6h
8	S	root	3065	3059	0	40	20	f626d040	1639	f62aab96	Feb 22	?	c
8	S	root	88	1	0	40	20	f5b8d708	377	f5b59df6	Feb 18	?	q
8	S	root	478	1	0	41	20	f5b8ec08	388	f5b5lbb8	Feb 18	?	0
8	S	root	94	1	0	41	20	f5b8d008	527	f5b59e46	Feb 18	?	d
8	S	root	150	1	0	41	20	f5da1a10	808	f5b59806	Feb 18	?	d
8	S	root	96	1	0	67	20	f5da2810	535	f5b59ad6	Feb 18	?	v
8	S	root	144	1	0	41	20	f5da0c10	2694	ef69f6lc	Feb 18	?	5d
8	S	root	1010	1	0	0	RT	f6lda330	496	f5dbeclc	Apr 12	?	n
8	S	root	106	1	0	41	20	f5da1310	485	f5b59e96	Feb 18	?	s
8	S	root	156	1	0	51	20	f5b8de08	446	f5b5lebB	Feb 18	?	n
8	S	root	176	1	0	53	20	f5da2110	740	f5b59036	Feb 18	?	d
8	S	root	129	1	0	56	20	f5d9fe10	447	f5b59cb6	Feb 18	?	d
8	S	daemon	130	1	0	41	20	f5d9f710	564	f5b59b76	Feb 18	?	d
8	S	root	14798	1	0	45	20	f5b8e508	616	f5b8e730	Mar 09	?	3d
8	S	root	133	1	0	51	20	f5e18818	507	f5b59c66	Feb 18	?	s
8	S	root	197	1	0	63	20	f5e15e18	284	f5e16040	Feb 18	?	d
8	S	root	196	1	0	41	20	f5da0510	429	f5c68f8e	Feb 18	?	c
8	S	root	213	1	0	41	20	f5e16518	586	f5c68b2e	Feb 18	?	4d
8	S	root	199	196	0	41	20	f5el6clB	451	f5b59f86	Feb 18	?	i
8	S	root	219	1	0	41	20	f5el7318	658	f5b59d06	Feb 18	?	m
8	S	root	209	1	0	41	20	f5e18118	234	f5c68e4e	Feb 18	?	d
8	S	root	2935	266	0	40	20	f6ldb130	2473	f62aaa56	Feb 22	?	4
8	S	root	16800	3059	1	81	30	f626f340	1466	f6lb345e	07:59:40	?	k
8	S	root	2963	2954	0	40	20	f5f52028	513	f6lb3l3e	Feb 22	?	e
8	S	root	479	1	0	55	20	f5ee7120	407	f5fde2c6	Feb 18	console	g
8	S	root	10976	1	0	65	20	f5f55828	478	f5c6853e	Jun 01	?	d
8	S	root	7468	1	0	46	20	f621da38	2851	8306c	Feb 24	?	d
8	S	root	266	1	0	41	20	f5ee5520	1601	f5c6858e	Feb 18	?	n
8	S	martyp	16763	16761	0	51	20	f6270140	429	f62701ac	07:46:46	pts/4	h

```
8  S   root 10995     1     0  41  20  f5b8f308  2350  f5fde5e6  Jun 01    ?     d
8  S   root   484   478     0  41  20  f5ee4e20   408  f5ee5048  Feb 18    ?     n
8  S   root   458     1     0  41  20  f5f54a28   504  f5fde906  Feb 18    ?     2m
8  O   root 16802 16763     1  61  20  f5ee7820   220            08:00:05  pts/4 1
8  S   root   471     1     0  41  20  f5f53c28   658  f5fde726  Feb 18    ?     d
8  S   root   474     1     0  51  20  f5f53528   804  f61a58b6  Feb 18    ?     g
8  S   root   485   458     0  40  20  f5f52e28   734  f607ecde  Feb 18    ?     74
8  S   root  2954  2936     0  40  20  f626e540   433  f626e5ac  Feb 22    ?     n
8  S   root  2936   266     0  66  20  fSee4720  1637  f5ee478c  Feb 22    ?     n
8  S   root  3061  3059     0  40  20  f5el7a18  2041  f6lb359e  Feb 22    ?     m
8  S   root  3058     1     0  40  20  f6ldaa30  1067  f62aadc6  Feb 22    pts/2 n
8  S   root   712   133     0  41  20  f6ld8e30   798  f6lb390e  Feb 18    ?     d
8  Z   root 11001 11000     0   0                                                >
8  S   root  2938     1     0  60  20  f5ee6320   513  f6olbfb6  Feb 22    ?     0
8  S   root  2999  2954     0  40  20  f621e138  1450  f6lb33be  Feb 22    pts/2 t
8  S   root  3059  3002     1  51  20  f626de40  4010  f62aafa6  Feb 22    pts/2 2n
8  S   root  3063  3059     0  50  20  f621e838  1952  f62aa556  Feb 22    ?
8  S   root  3099  3062     0  40  20  f5f52728  2275  f60a1d18  Feb 22    ?     0
8  S   root 11000 10995     0  48  20  f626d740  2312     55694  Jun 01    ?     e
8  S   root  3002  2999     0  43  20  f6ld8730   427  f6ldB79c  Feb 22    pts/2 =
8  S   root   730   133     0  40  20  f61d9530   422  f62aa9b6  Feb 18    ?     d
8  S   root  3062  3059     0  61  20  f62lb738  2275  f62aa506  Feb 22    ?     0
8  S   root  3067     1     0  40  20  f5ee5c20   424  f5ee5c8c  Feb 22    ?     d
8  S   root  3000     1     0  40  20  f6ld8030   518  f62aa8c6  Feb 22    ?     m
8  S   root  3078  3067     0  40  20  f6ld9c30   211  f5b5l2b8  Feb 22    ?     0
8  S   root 10984     1     0  41  20  f5f54328  2484  eee46e84  Jun 01    ?     lb
8  S   root 16761   133     0  44  20  f5ee4020   411  f5c6894e  07:46:45  ?     d
martyp $
```

이 실례에서 첫 렬은 F 기발이다. F 는 프로쎄스가 체계프로쎄스인 **핵심**(core)에로 전환되였는가를 비롯하여 여러가지 사실들에 대한 8 진수정보를 준다. 8 진수값은 때때로 체계마다 변하므로 현재체계에 대한 안내페지를 검사하여 기발들의 8 진수값을 보아야 한다.

S 는 상태를 보여 주기 위한 기발이다. 프로쎄스는 **휴식상태**(Sleeping), **대기상태**(Waiting), **실행상태**(Running), **중간상태**(Intermediate), **종결상태**(Terminated)에 있을수 있다. 또한 이 값들도 체계마다 변할수 있으므로 안내페지를 검사하여야 한다.

이 출력에서 많이 리용되는 정보는 좋은 값에 대한 NI, 프로쎄스의 기억자리 주소에 대한 ADDR, 프로쎄스의 물리적페지크기에 대한 SZ 그리고 프로쎄스가 기다리고 있는 사건에 대한 WCHAN 이다.

프로쎄스의 제거

ps 지령을 주어 어느 한 과제수행에서 벗어 나지 못하고 있는 프로쎄스를 발견한 다음 그 프로쎄스를 정지시키려는 경우에 kill 지령을 리용할수 있다. kill 지령은 지적된 프로쎄스에 신호를 보내는 봉사프로그람이다. 사용자는 kill 지령을 리용하여 자기의 프로쎄스를 제거할수 있다. 한편 상급사용자는 체계상의 거의 모든 프로쎄스들을 제거할수 있다.

프로쎄스를 제거하려면 kill 지령을 줄 때 프로쎄스 ID(PID)를 지적해 주면 된다. 아래의 실례에서는 martyp 가 소유하고 있는 모든 프로쎄스들을 찾기 위하여 ps 지령을 주고 한개 프로쎄스를 제거하며 그 프로쎄스가 실지로 제거되였는가를 검사하는 방법을 보여 주고 있다.

```
martyp $ ps -ef | grep martyp
    martyp  19336  19334  0  05:24:32  pts/4  0:01  -ksh
    martyp  19426  19336  0  06:01:01  pts/4  0:00  grep  martyp
    martyp  19424  19336  5  06:00:48  pts/4  0:01  find  /  -name  .login
martyp $ kill 19424
martyp $ ps -ef | grep martyp
    martyp  19336  19334  0  05:24:32  pts/4  0:01  -ksh
    martyp  19428  19336  1  06:01:17  pts/4  0:00  grep  martyp
[1] + Terminated            find  /  -name  .login  &
martyp $
```

이 실례에서는 martyp 가 소유하고 있는 프로쎄스 19424 를 제거하고 ps 지령을 다시 주어 그 프로쎄스가 실지로 제거되였는가를 확인한다.

kill 지령에 제거하려고 하는 모든 프로쎄스들의 번호를 공백으로 분리하여줌으로써 여러개 프로쎄스들을 지령행에서 제거할수 있다.

상급사용자로 가입되여 있는 경우에는 프로쎄스들을 제거할 때 특별히 주의하여야 한다. 그렇지 않으면 체계가 동작하는데 나쁜 영향을 줄수 있으며 프로쎄스들을 수동적으로 재실행하거나 체계를 재시동해야 할수 있다.

신호

kill 지령과 프로쎄스번호를 주면 kill 과 결합된 신호도 전송된다. kill 에 대한 실례에서는 신호에 대하여 규정하지 않았지만 15 또는 SIGTERM 이 기정신호로 리용되였다. 체계는 이 신호들을 리용하여 프로쎄스들과 통신한다. 프로쎄스를 종결시키는데 리용되는 신호 15 는 시동된 find 지령과 같은 사용자프로쎄스를 종결시키는데 충분한 쏘프트웨

어종결신호이다. 제거하기 어려운 프로쎄스는 SIGKILL 또는 신호 9 를 요구한다. 이 신호는 프로쎄스를 즉시 종결시킨다. SIGKILL 로 제거되는 프로쎄스들은 항상 원활하게 종결되지 않기때문에 이것을 마지막수단으로만 리용한다. 쉘과 같은 프로쎄스들을 제거하려면 때때로 SIGKILL 을 리용하여야 한다.

신호이름 또는 번호를 리용할수 있다. 신호번호들은 체계에 따라 변하므로 신호에 대한 안내페지에서 체계상의 신호목록을 보아야 한다.

가장 흔히 리용되는 일부 신호번호들과 그 번호들에 대응하는 신호들은 다음과 같다.

신호번호	신호
1	SIGHUP
2	SIGINT
3	SIGQUIT
9	SIGKILL
15	SIGTERM
24	SIGSTOP

이 책의 앞에서 설명한바와 같이 선택항목 "-s"를 **직결안내페지**(On-Line Manual Page)에서 제시된 구획번호와 함께 리용하면 규정된 구획의 내용을 볼수 있다. 실례로 5 구획에 있는 신호안내페지를 보려면 다음의 지령을 주면 된다.

```
$ man -s 5 signal
```

Linux 와 같은 UNIX 변종은 구획번호의 선택항목에서 대문자 S 를 요구한다.

id234 를 가진 프로쎄스를 SIGKILL 로 제거하려면 다음의 지령을 주어야 한다.

ps 와 kill 을 이 항목에서 취급한대로 리용하면 모든 UNIX 변종들에서 동작한다. 신호정의들은 UNIX 변종들마다 다르다는것을 명심하여야 한다. 어떤 UNIX 변종에는 pgrep 와 pkill 이라고 부르는 두개의 지령도 있다. pgrep 는 프로쎄스를 이름 또는 다른 특성에 따라 찾는다. 앞에서 사용자 martyp 가 소유한 프로쎄스들을 찾기 위하여 grep 와 결합된 ps 지령을 준 실례가 있다. pgrep 는 이 두 지령들의 결합으로서 규정되는 이름과 다른 특성들에 기초하여 프로쎄스들을 찾는다. 류사하게 pkill 은 규정되는 특성에 따라 프로쎄스를 식별하고 그것을 제거한다.

체계의 시동 및 끄기스크립트

UNIX 변종의 새 방안에 대한 시동 및 끄기스크립트들은 현재 시동 및 끄기스크립트들을 체계조성정보와 분리시키는 기술에 기초하고 있다. 체계가 시동 및 정지하는 방법을 변경하는데 스크립트를 수정하지 않아도 된다. 스크립트를 수정하는것은 일반적으로 위험한 일이다. 그대신 **구성변수**(configuration variable)들을 수정할수 있다. 시동 및 종결순서는 대부분의 UNIX 변종들에서 류사하게 적용하고 있는 **공업표준**(Industry Standard)에 기초하고 있다.

그러나 항상 한개의 UNIX 변종으로부터 다른 변종에로 넘어 갈 때에는 그 실현에서 약간한 차이가 생긴다. 여기서는 UNIX 체계상에서 실현된것과 거의 류사한 실현에 대하여 서술한다.

체계관리에서 시동과 끄기는 매우 중요하다. 응용프로그람들을 더 많이 적재하고 개성화할수록 시동과 끄기에 대한 지식이 더 많이 필요된다. 이 항목에서는 시동과 끄기를 개괄하고 체계를 끄는데 리용될수 있는 지령들을 취급한다.

다음의 성분들은 **시동 및 끄기모형**(Startup And Shutdown Model)에 있는것이다.

실행스크립트

실행스크립트들은 구성변수스크립트로부터 변수들을 읽고 시동 또는 끄기순서로 실행된다. 이 스크립트들은 보통 /sbin/init.d 에 배치되여 있다(어떤 체계들에서는 이 스크립트들을 /etc/init.d 에서 찾을수 있다.).

구성변수스크립트들

이것들은 부분체계를 가능하게 하거나 불가능하게 하는데 리용되는 변수들을 설정하거나 체계를 시동 또는 끌때 어떤 다른 기능를 수행시킬수 있는 파일들이다. 이 파일들은 /etc/rc.config.d 에 배치되여 있다(다른 체계들에는 이 파일이 존재하지 않는다.).

련결파일들

이 파일들은 스크립트들이 실행되는 순서를 조종하는데 리용된다. 이것들은 실지로 한 실행준위로부터 다른 실행준위에로 이행할 때 실행되여야 할 실행스크립트들에로의 련결이다. 이 파일들은 알맞는 실행준위에 대한 등록부 실례로 실행준위 0 에 대하여서는 /sbin/rc0.d, 실행준위 1 에 대하여서는 /sbin/rc1.d 등에 배치되여 있다. "S"로 시작되는 이 등록부의 파일들은 실행스크립트들이며 "K"로 끝나는 파일들은 끄기스크립트들이다(이 련결들은 어떤 체계들에서는 /sbin 대신에 /etc 에서 찾아 볼수 있다.).

련쇄스크립트

이 스크립트는 실행준위변환에 기초하여 실행스크립트들을 호출한다. 이 스크립트는 보통 /sbin/rc 이다.

그림 13-1 에서는 실행 및 끄기스크립트들에 대하여 흔히 리용되는 등록부구조를 보여 주고 있다.

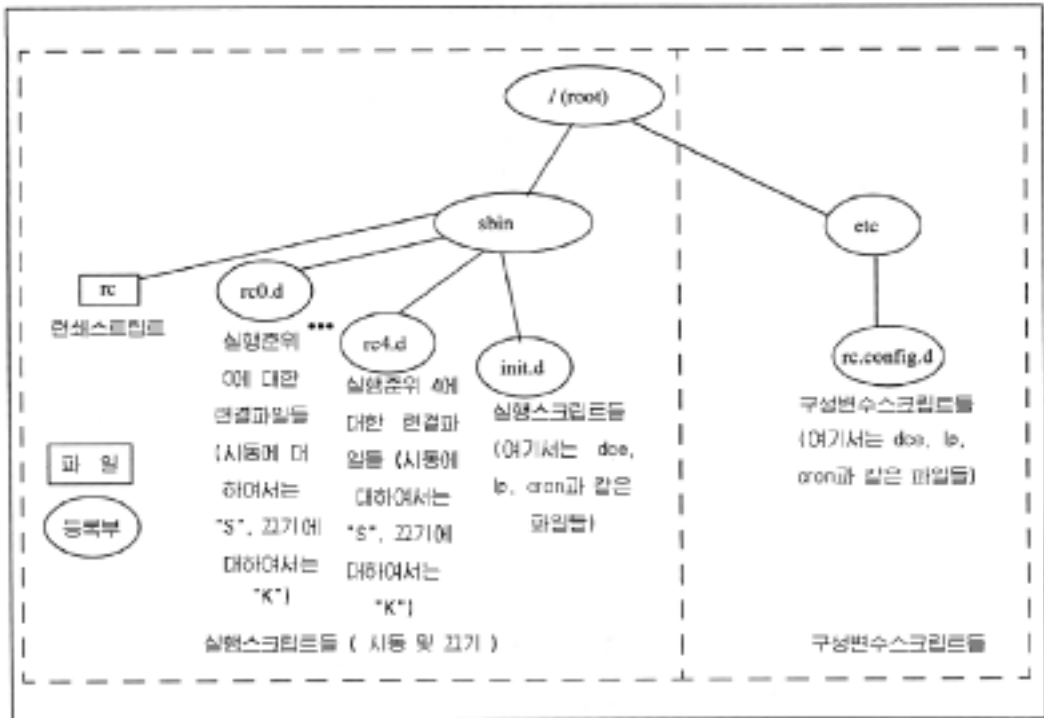



그림 13-1. 시동 및 끄기파일들의 조직

실행스크립트들은 시동 및 끄기과제들을 수행한다. /sbin/re 는 적당한 시동 또는 정지인수들을 가지고 실행스크립트를 호출한다. 이때 **콘솔**(console)우에서 적당한 시동 또는 정지통보문을 볼수 있다.

통보문들은 다음 세개 값들중의 하나를 가진다.

OK	이것은 실행스크립트가 정확히 시동 또는 꺼졌다는것을 지적한다.
FAIL	시동 또는 끌 때 문제가 발생하였다.
N/A	스크립트는 시동할수 있게 조성되지 않았다.

부분체계를 시동시키려면 /etc/rc.config.d 에 적당한 조성파일을 편집하여야 한다. /etc/rc.config.d/audio 를 보여 주는 아래의 실례에서는 변수 AUDIO_SERVER 가 1 로 설정되여 있다.

```
# * * * * * * *         File:  /etc/rc.config.d/audio       * * * * * * *
# Audio server configuration. See audio(5)
# AUDIO_SER:            Set to 1 to start audio server daemon
#
AUDIO_SERVER=1
```

이 실례의 결과로 체계가 시동될 때에는 다음과 같은 통보문이 나타나며

Start audio server daemon . . . . [OK]

체계가 종결될 때에는 다음과 같은 통보문이 나타난다.

Stopping audio server daemon . . . . OK

지금까지 실행준위에 대하여 여러번 언급하였다. 여기서 서술된 시동 및 끄기스크립트들뿐아니라 /etc/inittab 파일도 실행준위에 의존한다. 실행준위들은 UNIX 변종들에서 서로 차이난다. 일반적으로 보다 낮은 실행준위들은 낮은 준위기능을 위하여 리용되며 실행준위가 증가될 때에는 그 기능을 개선하여야 한다. 이 일반화가 항상 옳은것은 아니므로 리용되고 있는 UNIX 변종에 대하여 검사하여야 한다. 아래에서는 실행준위들의 목록과 그것들이 어떻게 리용되는가를 보여 주고 있다.

0	정지된 실행준위
s, S	단일사용자방식이라고도 부르는 실행준위 s 는 체계에 체계관리과제를 수행하는 사람이 하나밖에 없다는것을 담보한다. 일부 체계들에서 PROM **감시기준위**(Monitor Level)
1	실행준위 1 은 단일사용자방식에 대한 다양한 기초프로쎄스들을 시동한다.
2	실행준위 2 는 사용자들이 체계에 접근하는것을 허락한다. 이것을 다중사용자방식이라고도 부른다.
3	실행준위 3 은 NFS 파일체계를 반출하고 많은 UNIX 변종들에서 다른 자원들을 공유하기 위한것이다.
4	실행준위 4 는 공동탁상환경(HP CDE)을 포함하여 도형(graphics)관리프로그람을 일부 UNIX 변종들에서 시동시키며 다른 UNIX 변종들에서는 리용되지 않는다.
5	일부 UNIX 변종들에서는 리용되지 않으며 다른 UNIX 변종들에서는 "HALT"상태대신에 리용된다.
6	일부 UNIX 변종들에서는 리용되지 않지만 다른 변종들에서는 실행준위 3 으로 재시동하기 위하여 리용된다.

실행준위들은 UNIX 변종들에서 크게 차이나므로 자기가 리용하고 있는 UNIX 변종에 대한 실행준위들과 그것에 결합된 기능들을 검사하여야 한다.

/etc/inittab 은 실행될수 있는 다양한 프로쎄스들을 정의하는데도 리용되며 /sbin/init 에 의해서도 리용된다. 프로쎄스 /sbin/init 의 ID 는 1 이다. 이 프로쎄스는 체계에서 시동되는 첫 프로쎄스이며 선조프로쎄스를 가지지 않는다. 프로쎄스 init 는 체계의 실행준위를 결정하기 위하여 /etc/inittab 을 조사한다.

/etc/inittab 파일의 항목들은 다음과 같은 형식으로 되여 있다.

id:run state:action:process

id: 항목의 이름이다. id 는 4 개까지의 기호로 구성되며 파일에서 유일하여야 한다. /etc/inittab 에서 행앞에 "#"이 놓이면 설명문으로 취급된다.

run state: 지령의 실행준위를 규정한다. 한개이상의 실행준위가 규정될수 있다. 지령은 규정된 매 실행준위에 대하여 실행된다.

action: 11 개 작용들중 어느것이 프로쩨스 init 에서 일어 나는가를 정의한다. 11 개 작용들은 다음과 같다. initdefault, sysinit, boot, bootwait, wait, respawn, once, powerfail, powerwait, ondemand, off

process: 실행되여야 할 쉘지령(실행준위 및/또는 작용마당이 가리키는)이다.

/etc/inittab 의 한개 항목실례는 다음과 같다.

```
cons: 123456:respawn:/usr/sbin/getty console console
  |     |    |        |
  |     |    |        |>프로쩨스
  |     |    |>작용
  |     |>실행상태
  |>id
```

이것을 시동스크립트로 정의하지 않고 /etc/inittab 파일에 넣은 리유는 실행준위가 변하지 않는 한 콘솔은 제거될수 있으며 제거되자마자 재시동되여야 하기때문이다. respawn 은 프로쩨스들이 존재하지 않으면 어느한 프로쩨스를 시동하며 그것이 제거된후에 그것을 재시동한다. 이 항목은 모든 실행상태들을 보여 준다. 왜냐하면 콘솔이 항상 동작되여 있기를 바라기때문이다.

/etc/inittab 의 첫 행들중의 하나는 다음과 같을수 있다.

init:3:initdefault:

체계의 기정실행준위는 3 으로 정의되였다. 일부 UNIX 변종들에서 이 행은 다음과 같이 되여 있을수 있다.

is:3:initdefault:

이 두 실례들사이에는 실행준위의 id 만이 차이난다. 실행준위자체와 initdefault 는 같다. initdefault 는 init 가 처음으로 호출될 때 리용되는 작용이다.

시동과 끄기의 다른 방법

일부 UNIX 변종들에서는 시동 및 끄기방법이 약간 다르다. 그 절차를 설명하면 다음과 같다. 등록부 /sbin 는 rcS-rc6 과 같은 매 실행준위에 대한 파일을 포함한다. 이 파일들은 변수들, 검사조건들 그리고 봉사들을 시동하고 정지시키는 파일들에 대한 호출들을 포함한다. rcS-rc6 에 의해 실행되는 스크립트들은 /etc/rcS.d-/etc/rc6.d 와 같은 알맞는 실행준위등록부아래에 있다. 앞에서 서술한바와 같이 /etc/rcS.d-/etc/rc6.d 의 파일들중 S 로 시작되는것들은 프로쎄스들을 시동시키는데 리용되며 K 로 시작되는것들은 프로쎄스들을 정지시키는데 리용된다. 등록부 /etc/init.d 는 /etc/rcS.d-/etc/rc6.d 에 배치되여 있는 시동 및 제거스크립트에 의하여 리용되는 실행조종스크립트들을 포함한다. /etc/rcS.d-/etc/rc6.d 에 있는 스크립트들은 /etc/init.d 에 있는 프로그람들에로의 련결들인데 이것은 /etc/init.d 에 있는 조종스크립트들이 /etc/rcS.d-/etc/rc6.d 에서 /etc/init.d 에로의 련결창조에 의하여 리용된다는것을 의미한다. /etc/init.d 에는 다른 많은 등록부들에서 /etc/init.d 에 있는 스크립트들에 련결되여 리용되는 스크립트의 한개 방안이 있다. /etc/init.d 에 있는 조종파일들은 이 등록부로부터 프로그람들을 실행시키거나 정지시키는데 리용될수 있다. 실례로 lp 봉사를 정지시키려면 다음의 지령을 주어야 한다.

```
# /etc/init.d/lp  stop
```

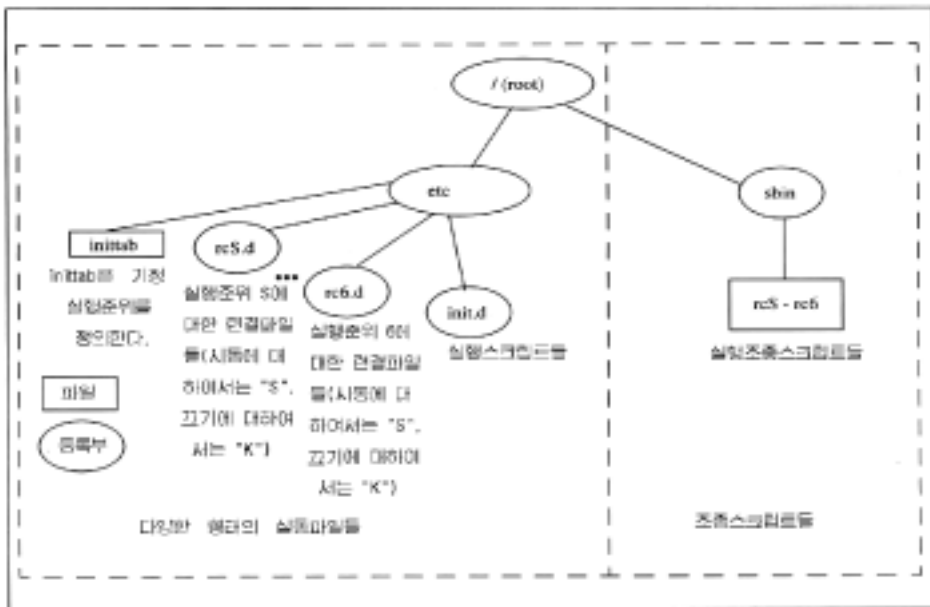



그림 13-2. 시동 및 끄기파일들에 대한 다른 조직

체계가 입장할 때 실행되여야 할 시동파일들은 /etc/inittab 로부터 이 시동방법으로 호출된다. /etc/inittab 는 initdefault 라고 부르는 기정실행준위와 적당한 실행파일들을 /sbin 에 정의한다. 그림 13-2 에서는 이 항목에서 론의된 시동 및 끄기파일들을 조직한것을 보여

주고 있다.

여기에서 서술된 체계시동 및 끄기의 기초개념들은 중요하다. 독자는 자기의 체계를 시동 및 끌수 있을것이며 여기서 서술된 일부 파일들을 수정할수 있을것이다. 그것들을 수정하기전에 시동 및 끄기파일들을 주의깊게 보아야 한다. 그 파일들과 절차들은 UNIX 변종들에서 크게 차이나므로 주의하지 않으면 위험해 진다.

체계끄기

체계를 끈다는것은 무엇을 의미하는가? 가장 단순한 형태로 그것은 shutdown 지령을 준다는것을 의미한다. shutdown 지령은 모든 프로쎄스를 종결시키는데 리용된다.

이 지령은 다음과 같은 선택항목들을 가진다.

-r	체계를 자동적으로 재시동한다. 이것은 일부 UNIX 변종들에서만 리용할수 있다.
-h	체계를 완전히 정지시킨다. 이것은 일부 UNIX 변종들에서만 리용할수 있다.
-y	아무런 질문도 제기하지 않고 체계를 종결한다.
grace 또는 -g	체계를 종결하기전에 파일을 보관하기 위한 사용자시간을 주고 응용프로그람을 탈퇴시키며 가입을 끝내기 위하여 기다려야 할 시간을 초수로 규정한다. 일부 UNIX 변종들에서 이것은 -g 이다.
-i	체계가 이행해야 할 상태를 규정한다.

체계를 종결하고 자동적으로 재시동하려면 다음의 지령을 주어야 한다.

```
$ shutdown -r
```

체계를 정지시키려면 다음의 지령을 주어야 한다.

```
$ shutdown -h
```

긴급한 체계끄기에 대하여 사용자에게 알려 주기 위한 통보문을 입력하겠는가고 물을수 있다. 통보문을 입력하면 즉시 모든 사용자에게 전달된다. 규정된 시간(기정으로는 60s)이 지난후에 체계는 종결프로쎄스를 시작한다. 체계가 정지되였다는 통보문을 받은후 모든 체계성분들의 전원을 끌수 있다. 아무런 질문도 제기하지 않고 체계를 2 분내에 끄려면 다음의 지령을 주어야 한다.

```
$ shutdown -h -y 120
```

때때로 shutdown 으로 단일사용자방식에 들어 가 **여벌복사**(backup)와 같은 어떤 과제를 수행하거나 론리기록권을 확장시킨다음 체계를 재시동하여 본래의 상태로 되돌아 갈 필요가 있군 한다. 단일사용자방식에 들어 가도록 체계를 끄려면 다음의 지령을 주어야 한다.

```
$ shutdown
```

이 지령은 UNIX 변종들에서 서로 차이난다. 일부는 상태규정을 허용하지 않으며 일부는 선택항목 -r 와 -y 를 지원하지 않고 그대신 상태를 규정해 줄것을 요구한다. 이 지령을 주기전에 모든 선택항목들에 대하여 잘 알아야 한다. 체계를 끄려고 할 때 체계를 리용하고 있는 다른 사람이 있는가를 알아 보아야 한다. 상급사용자접근을 요구하는 지령들을 리용할 때에는 매우 조심하여야 한다.

사용자와 그룹

체계관리자는 사용자를 설치하기전에 많은 선택항목들을 고찰해야 한다. 일단 설치된 사용자들에 대한 관리에 대하여서는 체계관리자가 관심을 돌릴 필요가 없다.

체계관리자는 자기가 설치한 매 사용자들에 대하여 사용자의 자료는 어디에 배치되여야 하는가? 누가 누구로부터 자료를 얻으려고 하는가?(이것으로부터 사용자들의 그룹를 정의한다.) 사용자들과 응용프로그람들은 어떻게 시동되여야 하는가, 사용자들이 좋아 할 쉘이 있는가를 결정하여야 한다. 그 다음 체계에서 고려해야 할 또 다른 문제는 도형사용자대면부의 개성화이다.

체계관리자가 가장 중요하게 고려해야 할 문제는 사용자자료와 관련되여 있다. 여러가지 리유로 하여 큰 체계를 관리하는데서 사용자자료를 재배렬하는것은 어려운 문제이다. 사용자자료는 한개 디스크에 다 채워 지는것이 아니다(일부 UNIX 변종들은 많은 디스크들을 포괄하는 기록권을 지원하는 기록권관리프로그람을 가지고 있다). 사용자들은 다른 사람의 자료들에 자유롭게 접근할수 없으며 한편 다른 사람의 자료에 너무 쉽게 접근하는것은 좋은 일이 아니다.

우의 질문들을 고찰하기전에 먼저 사용자를 첨가하는 기초적인 걸음들을 보기로 하자.

- 첨가하려는 사용자이름을 선택
- 사용자 ID 번호를 선택
- 사용자를 위한 그룹을 선택
- /etc/passwd 항목을 창조
- 사용자통과암호를 할당
- 사용자를 위한 홈등록부의 선택과 창조
- 사용자가 실행시키려는 쉘을 선택
- 사용자의 홈등록부에 시동파일을 배치
- 사용자가입급수를 검사

모든 사용자들에 대한 정보가 보관되여 있는 파일 /etc/passwd 에는 대부분 수행하여야 할 일들이 들어 있다. 그 항목들을 vipw 지령으로 /etc/passwd 파일에 만들어 넣을수 있다. 그림 13-3 에서는 /etc/passwd 의 본보기항목을 보여 주고 있다.

그림 13−3. 본보기 /etc/passwd 항목

매 마당들을 서술하면 다음과 같다.

이름 이것은 사용자이름이다. 이 이름은 체계상에서 사용자들이 상기하기 쉽게 정해 져야 한다. 전자우편을 보내거나 한 사용자로부터 다른 사용자들에게 파일들을 복사할 때 사용자이름을 상기하기 쉬울수록 더 좋다. 만일 한 사용자가 다른 체계상에서 어떤 이름을 가지는 경우에 해당 UNIX 체계상에서 동일한 사용자이름을 할당해주어야 한다. 일부 체계들은 친절하고 쉬운 사용자이름을 허락하지 않으므로 독자들은 낡은 체계와 결별하고 자기의 UNIX 체계상에서 의미가 좋고 상기하기 쉬운 사용자이름을 리용하기 시작하여야 한다. 사용자이름에는 안전성이 없다. 안전성은 사용자의 통과암호와 파일허락을 통하여 취급된다.

통과암호 이것은 코드화된 사용자의 통과암호이다. 매 마당에 하나의 별표(*)가 나타나면 가입급수는 리용될수 없다. 이 마당이 비였으면 사용자는 자기에게 할당된 통과암호를 가지지 않으며 자기의 이름만을 입력하여 가입할수 있다. 매 사용자는 자기의 통과암호를 주기적으로 변경시키는것이 좋다. 매 체계는 반드시 각이한 안전성을 가진다. 그러나 최소한 매 체계상의 매 사용자는 통과암호를 가져야 한다. 새로운 사용자를 설치할 때 그 사용자에게 처음으로 가입할 때의 통과암호를 참조할것을 요구할수 있다. 좋은 통과암호의 몇가지 특징들은 다음과 같다.

- 빗선(/), 점(.), 별표(*)와 같은 특수기호들을 포함할수 있는 최소 6개 기호들
- 단어들은 통과암호로 리용될수 없다.
- 사람의 이름, 주소, 이름난 체육단 등으로 통과암호를 만들지 말아야 한다.
- 123456 또는 qwerty와 같이 입력하기 쉬운것은 리용하지 말아야 한다.
- 맞춤법이 틀린 단어들은 일 없지만 그런것들은 리용하지 말아야 한다. 그런 단어들은 맞춤법검사프로그람들을 리용하여 알아 낼수 있기 때문이다.
- 리해하기 어려운 통과암호들을 생성하는 통과암호발생기가 가장 좋다.

사용자 ID(UID) 이것은 사용자의 식별번호이다. 체계상의 매 사용자는 유일한 UID를 가져야 한다. UID를 설정하는 방법은 UNIX 변종마다 서로 다른데 체계들의 사용자들에 대하여서는 100보다 작은 UID를 예약하여야 한다. 일부 UNIX 변종들은 일반 사용자들에 대한 UID를 1000에서 시작한다.

그룹 ID(GID) 그룹의 식별번호이다. 그룹의 성원들과 그것들의 GID는 파일 /etc/group에 있다. 체계관리자는 할당된 GID가 좋지 않으면 그것을 변경시킬수 있으나 그러자면 많은 파일들의 GID들을 변경시켜야 한다. 사용자가 파일을 창조할 때 그 파일에는 그 사용자의 UID뿐아니라 GID도 할당된다. 이것은 동일한 그룹의 사용자들이 많은 파일들과 등록부들을 창조한후에 체계관리자가 GID를 변경시키는 경우에는 그 모든 것들의 GID를 변경시켜야 한다는것을 의미한다. 체계그룹들에 대하여 10보다 작은 GID들을 보관한다.

선택적인 사용자정보 사용자의 전화번호 혹은 이름이 같은 항목들을 넣을수 있다. 이 자리를 비워 놓을수 있으나 체계 또는 망에 사용자들이 많은 경우에는 사용자의 이름과 확장자를 첨가하여 그 사용자와 접근해야 할 필요가 있을 때 그 정보를 함께 찾을수 있게 한다(이 마당을 가끔 GECO마당이라고 부른다).

홈등록부 홈등록부는 사용자의 모든 파일들과 등록부들을 위한 기정위치를 정의한다. 이것은 체계에 가입할 때의 현재작업등록부이다.

쉘 사용자가 체계에 가입할 때 실행시키려고 하는 시동프로그람이다. 쉘은 사용자가 지령행으로부터 주는 지령들에 대한 지령해석기이다. 체계관리자는 보통 개발된 실행파일들에 기초하여 무슨 쉘들이 지원되는가를 결정한다. 일반적으로 대부분의 쉘들이 체계상에서 지원된다.

사용자의 홈등록부위치는 파일 /etc/passwd에서 다른 하나의 중요한 항목이다. 사용자의 홈등록부위치는 파일들이 보관될 파일체계에서 선택되여야 한다. NFS와 같은 개선된 망기술이 존재하므로 사용자의 홈등록부는 그 사용자가 리용하고 있는 콤퓨터에 물리적으로 련결된 디스크에 있지 않아도 된다. UNIX 체계상에서 사용자의 홈등록부가 놓이는 전통적인 위치는 /home/martyp와 같은 /home 등록부이다.

등록부 /home에는 짧은 시간에 많은 내용들이 기입된다. 사용자는 자기의 홈등록부에서 파일들을 보통의 방법으로 창조하고 제거하므로 뿌리파일체계나 응용프로그람령역과 같은 정적령역에서보다 사용자령역에서 더 많은 일을 하게 된다. UNIX와 응용프로그람들을 적재한 다음에는 파일과 등록부의 첨가 및 삭제를 비교적 적게 수행하여야 한다. 사용자령역은 유지하기 힘들 정도로 련속적으로 갱신된다.

passwd 파일은 보통의 방법으로 날자를 계산한다. 사용자들은 체계에 드나들면서 이 파일을 계속 변경시킨다. 대부분의 UNIX 변종들에는 passwd 파일이 있는가를 검사하는 pwck라고 부르는 프로그람이 있다. 흥미 있는것은 사용자만이 아니라 체계관리자도 이 프로그람을 리용할수 있다는것이다. 아래의 실례에서는 기정통파암호파일상에서 pwck가 수행된 결과를 보여 주고 있다.

martyp $ **pwck**

webuser:x:80:80::/usr/local/etc/httpd/htdocs:/bin/sh

```
        Login directory not found
teacher:x:8057:80:Desktop  Classroom Teacher:/usr/local/etc/pwserver/www/:/bin/kh
        Login directory not found
oracle:x:200:201:Oracle User:/home/oracle:/bin/ksh
        Login directory not found
opc_op:x:777:77:OpC default operator:/export/home/opc_op:/bin/sh
        Login directory not found
denise - Login name not found on system
martyp $
```

이 실례는 여러 사용자들에 대한 가입등록부가 존재하지 않는다는것을 보여 준다. 가입등록부가 존재하지 않는것은 문제로 되지 않는다. 왜냐하면 일부 응용프로그람들은 가입이름은 요구하지만 홈등록부는 요구하지 않기때문이다. 다른 한편 가입등록부는 제거하고 passwd 에 있는 항목들은 제거하지 않으면 사용자들은 가입등록부가 없는 체계상에서 작업하게 될수 있다. 이 실례의 마지막문제는 사용자 denise 와 같이 체계상에서 파일 또는 등록부를 가지지 않는 사용자에 관한 문제이다. pwck 는 규정된 통과암호파일(기정으로 passwd)에서 다음과 같은 정보를 유효하게 만든다.

- 정확한 마당의 개수
- 가입이름
- 사용자 ID
- 그룹 ID
- 이미 존재하는 가입등록부
- 이미 존재하는 사용자의 기정쉘

pwck 는 적게 리용되지만 유용한 프로그람이다. 체계관리자들을 위하여 체계가 정상인가를 자주 검열해야 하는데 이때 이 pwck 프로그람을 리용할수 있다. pwck 가 오유 또는 경고를 찾지 못했을 때에는 응답을 하지 않는다.

사용자들을 그룹에 할당

사용자와 관련된 모든 정보들을 정의한후에 체계관리자는 그룹을 고찰하여야 한다. UNIX 환경에서 체계관리자는 자기의 모든 사용자들이 비록 각이한 그룹에 있다고 해도 동일한 그룹에 있다는것을 찾을 때까지 그룹들을 찾는다. 그룹을 취급하기전에 사용자에게 속하는 파일과 파일에 대하여 정의된 접근방법을 고찰하자.

```
$ ls -1
-rwxr-x--x 1 donna users 120 Jul 26 10:20 sort
```

체계상의 매 파일에 대하여 UNIX는 다음과 같은 3가지 접근방법을 지원한다.

- 사용자접근(u). 그 파일의 소유자에게 허락된 접근
- 그룹접근(g). 파일의 소유자가 들어 있는 그룹의 성원들에게 허락된 접근
- 기타접근(o). 모든 사람들에게 허락된 접근

이 접근권한들은 지령을 줄 때 허락을 r(읽기), w(쓰기), x(실행)에 설정하여 정의한다. 우의 용지(ls -l)에 대하여 표 13-1에 있는 허락을 줄수 있다.

표 13-1 허 락 들

접근	사용자접근	그룹접근	기타
읽기	r	r	-
쓰기	w	-	-
실행	x	x	x

접근권한들은 세개씩 그룹에 배렬되여 있다. 세개 허락그룹에 대한 세개 접근준위가 각각 존재한다. 소유자(이 경우에 donna)는 파일상에서 읽기, 쓰기, 실행할수 있다. 그룹 users의 성원들은 파일의 읽기 및 실행접근으로 허락되여 있다. 다른 사람들은 실행접근만으로 허락되여 있다.

이 허락들은 사용자들을 그룹에 배렬할 때 중요하게 고찰되여야 한다. 여러 사용자들이 동일한 파일에로의 접근을 요구한다면 체계관리자는 그 사용자들을 동일한 그룹에 넣으려고 한다. 이 그룹안의 모든 사용자들에게 파일들에로의 rwx 접근을 할수 있으나 그렇게 하면 여러 사용자들은 어떤 파일을 편집할수 있고 다른 사용자들은 그것을 몰라 혼란이 생길수 있게 된다. 다른 한편 한개 파일을 여러개 복사하여 매 사용자가 자기의 사본을 가지게 할수 있으나 그렇게 하면 파일에 대하여 여러개 방안들이 있게 된다. 가능하면 사용자들을 그들의 작업에 기초하여 그룹에 할당하여야 한다.

파일 /etc/group은 그룹이름, 코드화된 통과암호(드물게 리용되는), 그룹 ID 그리고 그룹안의 사용자들을 포함한다. /etc/group 파일의 한가지 실례를 보면 다음과 같다.

```
root::0:root
other::l:root, hpdb
bin::2:root, bin
sys::3:root, uucp
adm::4:root, adm
daemon::5:root, daemon
mail::6:root
lp::7:root, lp
tty::10:
```

nuucp::11:nuucp
military::25:jhunt，tdolan，vdallesandro
commercial::30:ccascone，jperwinc，devers
nogroups:*:-2:

이 파일 /etc/group 은 두개의 각이한 사용자그룹을 보여 준다. 비록 모든 사용자들이 **탁상출판도구**(Desktop Publishing Tool)와 같은 동일한 응용프로그람을 실행시키지만 일부 사용자들은 "상업적"제품에 대한 문서상에서 작업하며 일부 사용자들은 "군사적"인 문서상에서만 작업한다. 이것은 체계관리자로 하여금 두개의 그룹 즉 상업적문서준비를 위한 그룹과 군사적문서준비를 위한 그룹을 창조하게 한다. 한 그룹의 모든 성원들은 현재 쓰이는 문서가 무엇인가를 알고 다른것의 작업과 그 중요성을 존중한다. 한 그룹에 속하는 성원들은 서로 무엇을 하고 있는가를 알고 있으며 삭제되지 말아야 할 파일들을 알고 있으므로 체계관리자는 그들에 대하여 별로 관심하지 않아도 된다. 그러나 모든 사용자들을 한개 그룹에 넣으면 파일들을 보관하는데 많은 시간이 랑비된다. 왜냐하면 이 넓은 그룹의 사용자들은 자기 그룹의 다른 성원들이 소유하고 있는 파일들을 찾지 못하기때문이다. 사용자들은 newgrp 지령으로 그룹을 변경시킬수 있다.

group 파일도 보통 방법으로 날자를 계산한다. 사용자들이 체계에 드나들 때마다 passwd 와 group 파일들에 대한 변경이 요구된다. 대부분의 UNIX 변종들에는 group 파일들이 있는가를 검사하는 grpck 라는 프로그람이 있다. 흥미 있는것은 이 프로그람을 사용자들만이 아니라 체계관리자들도 자주 리용한다는것이다.

아래의 실례에서는 통과암호를 위해 리용되는 기정파일상에서 grpck 가 실행되는것을 보여 주고 있다:

```
martyp $ grpck

database:10:
        No users in this group

developmentl:*:200:nadmin，charles，william
        william - Login name not found in password file

bin::2:root，bin，daemon
        bin - Duplicate logname entry（gid first occurs in passwd entry）

sys::3:root，bin，sys，adm
        sys - Duplicate logname entry（gid first occurs in passwd entry）

adm::4:root，adm，daemon
        adm - Duplicate logname entry（gid first occurs in passwd entry）

uucp::5:root，uucp
        uucp - Duplicate logname entry（gid first occurs in passwd entry）
```

```
tty::7:root, tty, adm
      tty - Logname not found in password file
lp::8:root, lp, adm
      lp - Duplicate logname entry (gid first occurs in passwd entry)
nuucp::9:root, nuucp
      nuucp - Duplicate logname entry (gid first occurs in passwd entry)
martyp $
```

이 실례는 이 체계에 파일 group 와 관련된 다양한 문제들이 있다는것을 보여 준다. 그 문제들은 그룹자료기지에 사용자들이 없는것, passwd 에 항목을 가지지 않는 그룹에 어떤 사용자가 있는것 그리고 여러개의 중복되는 항목들이다. 이것들의 일부는 문제로 되지 않을수도 있다. 그러나 grpck 에서 제기되는 문제는 체계에서는 제거되였으나 그룹에서는 제거되지 않은 사용자에 관한 문제이다. grpck 는 규정된 그룹파일(기정으로는 group 이다.)에 있는 다음의 정보를 유효하게 한다.

- 정확한 마당의 개수
    - 그룹이름
    - 그룹 ID
    - 그룹의 최대 개수를 넘어 서는 가입이름
    - 통과암호파일에서 나타나는 가입이름

grpck 는 적게 리용되는 유용한 프로그람이다. 체계관리자들을 위하여 그들의 체계가 정상인가를 자주 검열하여야 하는데 이 프로그람을 안정성검사에 리용할수 있다.

디스크와 관련된 개념

체계관리자는 디스크와 파일체계를 설치하고 관리하며 감시하는데 많은 시간을 소비한다. 여기서는 기초적인 내용만을 취급하려고 한다. 사용자들은 디스크, 파일체계 등을 설치하는데 리용되는 지령들을 리용하지 못하게 되여 있으므로 설치와 관련한 내용은 취급하지 않겠다. 초기의 디스크 및 파일체계설치와 관련되는 지령들은 newfs, dd, fsck, mknod 등이다. 이 지령들은 UNIX 사용자지도서에 서술되여 있다. 이 항목에서는 다음과 같은 다양한 내용 즉 파일체계, 교환공간, 일부 설치파일, 망파일체계(NFS)들을 취급한다.

장비된 파일체계들과 교환공간의 보기

파일체계들에 대하여 흥미를 가질 때 수행하는 첫 작업들중의 하나는 현재 자기의 체계에 어떤 파일체계들이 장비되여 있는가 하는것과 그것들의 특성들을 알아 보는것이다. df 지령은 장비된 파일체계들과 매 파일체계상에서의 용량과 관련된 정보들을 렬거해 준다. 아래에서는 Solaris 체계에서의 df 출력에 대하여 보여 주고 있다.

```
martyp $ df
/proc            (/proc              ):        0   blocks      927  files
/                (/dev/dsk/c0t3d0s0  ):  2284464   blocks   438864  files
/dev/fd          (fd                 ):        0   blocks        0  files
/tmP             (swap               ):   116584   blocks    10390  files
/home/ptc-nfs    (ptc-nfs:/export/users2/home):22467786   blocks    -1  files
/opt/local       (ptc-nfs:/export/opt/local):1292902   blocks       -1  files
martyp $
```

이 출력은 df 지령을 선택항목들이 없이 주었을 때 제시된다. 현재 설치되여 있는 모든 파일체계들은 자기의 용량정보와 함께 렬거되여 있다. df 지령을 줄 때 선택항목 -t를 주면 총체적인 파일체계들을 생성하며 대부분 체계들상의 **교환공간**(Swap Space)에 대한 출력을 준다. 아래의 실례에서는 df -t가 실행되는 프로쎄스를 보여 주고 있다.

```
martyp  $  df  -t
/proc            (/proc              ):        0   blocks      927  files
                                 total:        0   blocks      988  files
/                (/dev/dsk/c0t3d0s0  ):  2284464   blocks   438864  files
                                 total:  3806172   blocks   476288  files
/dev/fd          (fd                 ):        0   blocks        0  files
                                 total:        0   blocks       66  files
/tMP             (swap               ):   116600   blocks    10390  files
                                 total:   197104   blocks    10455  files
/home/ptc-nfs    (ptc-nfs:/export/users2/home):  22448064  blocks   -1  files
                                 total:139264000   blocks       -1  files
/opt/local       (ptc-nfs:/export/opt/local):1292900   blocks       -1  files
                                 total:  31866880  blocks       -1  files
martyp  $
```

이 출력에 들어 있는것들은 총체적으로 교환공간과 관련되는것들이다. 교환공간은 체계에서 매우 중요한 개념이다. 체계의 실지 기억(RAM)이 전부 소비될 때 어떤 정보가

교환공간령역에 들어 가는데 이 정보는 UNIX 변종들에서 서로 다르다. 일부 UNIX 변종들은 교환공간에 아무런 일도 하지 않는 프로쎄스들 및 그것들과 관련된 모든 정보들을 옮겨 놓는다. 다른 변종들은 아무런 일도 하지 않는 정보블로크들만을 옮겨 놓는다. 중요한것은 교환공간이 이 목적에 리용되는 디스크상의 구획 또는 기록권이라는것이다.

체계관리자에게 있어서 가장 어려운 과제들중의 하나는 체계상에서 요구되는 교환공간의 크기를 결정하는것이다. 흔히 리용되는 엄지손가락의 규칙은 교환공간의 크기를 RAM의 크기의 두배로 되게 한다.

대부분의 체계들은 체계가 헝클어 졌을 때 RAM의 내용을 꺼내 보는데 리용되는 **쏟기공간**(Dump Space)도 가진다. 체계가 헝클어 졌을 때 RAM의 모든 내용들을 디스크에 기입하는데는 오랜 시간이 걸린다. 일부 UNIX 변종들은 리용되였던 RAM의 령역만을 체계가 헝클어 진 시각에 쏟기공간에 기입하여 체계를 되살리고 보다 더 빨리 실행될수 있게 한다.

df 실례를 다시 보면 선택항목 -t는 교환공간를 포함하여 파일체계의 모든것을 보여준다.

교환공간과 특별히 관련된 정보는 swap지령으로 볼수 있다.

아래의 두 실례들에서는 교환공간에 대한 정보를 렬거하도록 선택항목 -l을 준것과 교환공간의 내용을 볼수 있도록 선택항목 -s을 준 swap지령들을 보여 주고 있다.

```
martyp $ swap -l
swapfile            dev     swaplo  blocks  free
/dev/dsk/c0t3d0s1   32,25   8       263080  74200

martyp $ swap -s
total: 99104k bytes allocated + 10200k reserved = 109304k
used, 63252k available
martyp $
```

장치파일들에 대하여서는 이 장에서 후에 취급된다. 이 실례를 취급하는 목적은 교환공간을 위하여 리용되는 /dev/dsk/c0t3d0s1이라고 부르는 디스크의 부분령역이 있다는것을 보여 주자는데 있다. swap -l로 보여 준바와 같이 디스크 c0t3d0에는 교환공간을 위하여 예약된 s1이라고 부르는 부분령역이 있다.

swap는 교환공간을 첨가하고 제거하는데도 리용될수 있다. 교환공간의 첨가와 제거는 체계관리자가 해야 할 일이기때문에 이 사실에 대하여서는 취급하지 않겠다.

장비된 파일체계들은 임의의 시각에 mount 지령으로 볼수 있다. 이 지령은 보통 사용자들이 장비된 파일체계들을 보는데 리용되지만 사용자로서는 파일체계들을 그 어떤 방법으로도 변경시킬수 없다. 이 책에서 취급되는 대부분 지령들의 결과들은 UNIX 변종들에서 어느정도 차이난다. mount지령도 례외로 되지 않는다.

다음의 실례들에서는 각각 Solaris, AIX, HP-UX에서 mount 지령을 준것을 보여 주고 있다.

```
sunl $ mount
/proc on /proc read/write/setuid on Thu Feb 18 11:09:14
/ on /dev/dsk/c0t3d0s0 read/write/setuid/largefiles on Thu Feb 18 11:09:14
/dev/fd on fd read/write/setuid on Thu Feb 18 11:09:14
/tmp on swap read/write on Thu Feb 18 11:09:16
/opt/local on ptc-nfs:/export/opt/local read/write/intr/soft/remote on Sun Jul 9
sunl $

ibmi $ mount
 장비된        장비된                      vfs      날자    선택항목
 마디점         위치
-------------------------------------------------------------------------------------
/dev/hd4       /                          jfs      Jul 02  19:02  rw, log=/dev/hd8
/dev/hd2       /usr                       jfs      Jul 02  19:02  rw, log=/dev/hd8
/dev/hd9var    /var                       jfs      Jul 02  19:02  rw, log=/dev/hd8
/dev/hd3       /tMP                       jfs      Jul 02  19:02  rw, log=/dev/hd8
/dev/hd10      /usr/sys/inst.images       jfs      Jul 02  19:03  rw, log=/dev/
/dev/lv00      /opt                       jfs      Jul 02  19:03  rw, log=/dev/hd8
/dev/bakup     /home.bak                  jfs      Jul 02  19:03  rw, log=/dev/log10
/dev/cd0       /usr/lpp/info/data/techlib cdrfs    Jul 02  19:03  ro
/dev/lv01      /root/users                jfs      Jul 02  19:03  rw, log=/dev/hd8
-hosts         /net                       autofs   Jul 02  19:04  ignore
auto.users     /users                     autofs   Jul 02  19:04  ignore
auto.ptc       /ptc mnt                   autofs   Jul 02  19:04  ignore
auto.indirect  /local-mnt                 autofs   Jul 02  19:04  ignore
auto.direct    /home/ptc-nfs              autofs   Jul 02  19:04  ignore
auto.direct    /opt/local                 autofs   Jul 02  19:04  ignore

ibml $

hpl 22: mount
/ on /dev/vg00/lvol3 log on Wed Jun 16 11:20:07
/stand on /dev/vgo0/lvol1 defaults on Wed Jun 16 11:20:10
/var on /dev/vgo0/lvol8 delaylog on Wed Jun 16 11:20:21
/usr on /dev/vgo0/lvol7 delaylog on Wed Jun 16 11:20:21
/tmp on /dev/vgo0/lvol6 delaylog on Wed Jun 16 11:20:22
/opt on /dev/vgo0/lvol5 delaylog on Wed Jun 16 11:20:22
/home on /dev/vgo0/lvol4 delaylog on Wed Jun 16 11:20:22
hpl 23:
```

시동될 때 장비되는 파일체계들은 대부분 UNIX 변종들에서 /etc/vfstab 또는 /etc/fstab 에 있다. 이 파일은 장비될 장치들, 장비될 위치, 파일체계형식, 선택항목들과 관련되는 정보들과 기타 다른 중요한 정보들을 포함한다. 체계에 있는 이 파일을 보고 체계관리자가 기정으로 설치하는 파일체계가 무엇인가 하는것을 알수 있다.

디스크리용의 결정

체계관리자들은 체계상에서 사용자들, 응용프로그람들, 그룹들 등에 의하여 소비되는 디스크공간의 량을 알려고 한다. 지령 du 로 이것을 결정할수 있다. 파일의 디스크리용정형을 지령 du 로 볼수 있다. 홈등록부 martyp 에 대해서는 지령 du 를 다음과 같이 준다.

```
martyp $   du
2          ./test
60         ./shellprogs
1366       ./.trash
6024       .
martyp $
```

이 출력은 이 등록부가 6024Kbyte 또는 약 6Mbyte 를 소비한다는것을 보여 준다. 지령 du 에 선택항목 -s 를 붙여 주면 home 등록부전체에 대한 다음과 같은 결과를 얻는다.

```
martyp $ du -o /home/nfs/*
0          /home/nfs-home/Mail
0          /home/nfs-home/SomeResults
1          /home/nfs-home/admin
30285      /home/nfs-home/achil
18079      /home/nfs-home/admin1
12         /home/nfs-home/aguaris
1262210    /home/nfs-home/alfonso
8          /home/nfs-home/amand
395838     /home/nfs-home/andre
28         /home/nfs-home/andrej
94448      /home/nfs-home/annelora
605738     /home/nfs-home/annetest
15         /home/nfs-home/badmin
448544     /home/nfs-home/barryworgo
7          /home/nfs-home/bbnuser
0          /home/nfs-home/bdonalla
```

```
0               /home/nfs-home/benilla
2565325         /home/nfs-home/benranos
3    /home/nfs-home/besinerro

                ⋮

273936          /home/nfs-home/thalla
6814            /home/nfs-home/timk
8               /home/nfs-home/tobbaa
298160          /home/nfs-home/tomphon
20              /home/nfs-home/tompsoca
5856            /home/nfs-home/vanhall
657543          /home/nfs-home/verdera
79248           /home/nfs-home/vobollas
24552           /home/nfs-home/waldok
martyp  $
```

이것은 큰 등록부를 관리하는데 편리한 기능이다. 사용자들은 자기가 소비하는 디스크공간의 량을 알려고 하는 경우에 이 지령을 자기의 홈등록부상에서 주어야 한다.

체계여벌복사

체계관리자들은 **체계여벌복사**(Backup)와 **회복**(Recovery)에 많은 시간을 소비한다. 이 항목에서는 여벌복사와 회복에 대한 일부 개념들을 서술하고 그다음 대부분 UNIX 변종들에서 볼수 있는 몇가지 여벌복사지령들을 취급한다. 이 지령들은 대부분의 체계들에서 리용가능하기때문에 널리 리용되고 있다. UNIX 환경들에는 개선된 기능을 제공하는 체계여벌복사와 회복프로그람들이 있다.

우선 체계여벌복사에 대하여 고찰한다.

여벌복사는 체계와 관련된 문제를 해결하는 수단이다. 체계와 관련된 문제에는 디스크상의 모든 자료바이트들을 파괴시키는 디스크하드웨어문제로부터 시작하여 실지로 필요되는 파일을 우연히 삭제해 버리는 사용자에 관한 문제들이 속한다. 디스크하드웨어문제는 가장 어려운 문제인데 이것을 해결하기 위해서는 체계를 전부 여벌복사시켜야 한다. 사용자에 관한 문제는 **정규여벌복사**(Regular Backup)와 **증가여벌복사**(Incremental Backup)로 회복할수 있다. 이것은 전면적인 체계여벌복사를 위하여 정규여벌복사와 증가여벌복사를 가능한껏 자주 수행하여야 한다는것을 의미한다.

여기에서는 흔히 리용되는 여벌복사프로그람들을 간단히 개괄한다.

tar   tar 는 가장 널리 리용되는 여벌복사프로그람이다. tar 테프에는 많은 응

용프로그람들이 들어 있다. 이것은 다른 UNIX 체계와의 자료교환에서 가장 널리 리용되는 형식이다. tar 는 가장 오랜 UNIX 여벌복사방법이므로 모든 UNIX 체계상에서 실행된다. tar 테프의 끝에는 파일들을 첨가할수 있다. 다른 UNIX 사용자에게 파일들을 보낼 때에도 tar 를 리용한다. tar 는 속도가 빠르지 못하기 때문에 **전면여벌복사**(Full Backup) 또는 증가여벌복사에 리용되지 않고 있다. 파일들을 tar 로 테프우에 적재하고 그것들을 다른 체계상에 보관할 때 본래의 사용자들과 그룹들은 유지된다. 실례로 frank 에 속하는 모든 파일들을 되살리고 그것들을 다른 체계에 적재하려면 다음의 지령들을 리용하여야 한다.

```
$ cd/home/frank
$ tar-cvf/dev/rmt/0m.
```

선택항목 c 는 새로운 파일 tar 을 창조하고 선택항목 v 는 불필요한 출력을 생성하며 선택항목 f 는 여벌복사에 리용되는 장치파일 /dev/rmt/0m 을 규정한다. 점은 여벌복사가 현재작업등록부에서 시작된다는것을 가리킨다. 그러면 frank 와 그 그룹이 체계상에 아직 존재하지 않아도 frank 의 파일들을 다른 체계상에 적재할수 있다.

cpio
: cpio 도 tar 처럼 편리하고 리용하기 쉽다. 또한 cpio 는 tar 보다 훨씬 빠르다. cpio 는 등록부나무를 반복시키는데 좋다. cpio 는 앞에서 론의된 증가여벌복사를 지원한다. cpio 에 파일들의 목록을 주면 여벌복사를 수행한다.

dd
: 이것은 비트별 복사이다. 이 지령은 파일들과 소유기록권을 복사하지 않고 비트들만을 복사하므로 그리 좋지 못하다. 그러므로 tar 혹은 cpio 로 할수 있었던 dd 테프로부터 파일들을 선택할수 없다. dd 는 현재디스크로부터 자료를 다른 디스크에 복사하여 첫 디스크의 **반사영상**(Mirror Image)을 창조하는데 널리 리용된다.

dump 와 ufsdump
: dump 와 ufsdump 는(UNIX 변종에 의존하여) 전면여벌복사 또는 증가여벌복사를 생성하기 위한 프로그람이다. 증가여벌복사의 형을 규정하는 **쏟기준위**(Dump Level)를 규정할수 있다. 이 지령으로 규정할수 있는 여벌복사의 준위들은 10 개이다. 준위 0 은 가장 낮은 준위이며 전면여벌복사이다. 또한 쏟기기록을 갱신하며 전면여벌복사와 증가여벌복사가 계속 수행되도록 할수 있다. 파일들은 UNIX 변종이 어느 지령을 지원하는가에 따라 후에 restore 또는 ufsrestore 로 회복될수 있다.

크론일감들의 일정작성

크론데몬(Cron Daemon)을 리용하여 과제들이 주기적으로 실행되도록 일정을 세울수 있다. 크론데몬은 체계가 실행될 때 실행하여 계속 실행된다. 크론은 조성파일들을 읽고 그 내용에 작용한다. 전형적인 조성파일에는 실행해야 할 지령과 그 지령을 실행한 날자와 시간 그리고 그 지령을 실행하는 사용자이름이 있다. 이 일정을 어느한 규정된 시간에 지령들을 주는 방법으로 볼수 있다. 조성파일들을 crontab 파일이라고 부른다.

crontab 파일은 cron 에 의하여 자동적으로 실행되는 일감들의 일정을 수립하는데 리용된다. crontab 파일은 보통 등록부 /var/spool/cron/crontabs 에 있다. 아래의 실례들이 실행된 Red Hat Linux 체계는 /var/spool/cron 에 있다.

crontab 파일에 있는 항목들의 형식은 다음과 같다.

minute hour monthday month weekday username command

minute	시간의 분, 0-59
hour	하루에서의 시간, 0-23
monthday	월에서의 날, 1-31
month	년에서의 달, 1-12
weekday	주에서의 날, 0(일요일)-6(토요일)
user name	필요할 때 지령을 실행시키려고 하는 사용자(실례에서는 리용되지 않는다.)
command	실행하려는 지령행 또는 스크립트파일.

crontab 항목들이 동일한 형식과 동일한 순서로 되여 있다는것을 보기 위하여서는 UNIX 변종을 검사하여 보아야 한다.

과제를 수행하기 위하여 crontab 파일에서 minute, hour, monthday, month, weekday 를 규정하는데는 많은 선택항목들이 리용된다. 한개 항목을 한개 마당과 그 뒤에 공백, 여러개 항목들을 반점으로 분리하여 임의의 마당에, 두개 항목들을 범위를 가리키는 기호(-)로 분리시키거나 그 마당에 대한 모든 가능한 항목들을 대응시키는 별표(*)로 렬거할수 있다.

cron 이 어떻게 작업하는가를 보기 위하여 아주 단순한 실례를 창조할수 있다. 다음의 내용들을 가지고 있는 listing 이라고 부르는 파일을 홈등록부에 창조하려고 한다. (Linux 체계에서 /root):

```
* * * * * ls ?l / > /root/listing.out
```

이 파일은 매 분마다 뿌리등록부의 긴 목록을 생성하고 홈등록부에 있는 listing.out 에 출력을 보낸다. crontab 를 기동시키거나 설치하려면 간단히 crontab 지령과 그 파일의 이름을 준다. 또한 그 파일을 어느한 규정된 사용자에게 결합시키려면 사용자이름을 규

정할수 있다. crontab 파일을 설치한 후에 crontab -l 을 주어 처리된 crontab 파일을 볼수 있다. 아래의 실례에서는 listing 이라고 부르는 crontab 파일을 가지고 작업한 프로쎄스를 보여 주고 있다.

```
[root@linuxl /root]# cat listing
* * * * * ls -1 / > /root/listing.out
[root@linuxl /root]# crontab listing
[root@linuxl /root]# crontab -1
# DO NOT EDIT THIS FILE - edit the master and reinstall.
#（listing installed on Fri Aug 6 10:40:03 ）
#（Cron version   $Id: crontab.c, v 2.13 03:20:37 vixie Exp $）
* * * * * ls -1   / > /root/listing.out
[root@linuxl /root]# ls -l

total 640
-rw -------     1   root  root   647168    Aug    5 23:22 core
-rw -------     1   root  root   7         Aug    6 10:14 dead.letter
-rw-r--r--      1   root  root   39        Aug    6 10:16 listing
-rw-r--r--      1   root  root   909       Aug    6 10:41 listing.out
-rw-------      1   root  root   516       Aug    6 10:15 mbox
-rw-r--r--      1   root  root   0         Aug    6     10:39 typescript
[root@linuxl /root]# cat listing.out
total 1182
drwxr-xr-x      2   root  root   2048      Jun    18 19:38   bin
drwxr-xr-x      2   root  root   1024      Jun    18 19:41   boot
-rw-------      1   root  root   1138688   Aug    5 23:21    core
drwxr-xr-x      5   root  root   34816     Aug    6 09:21    dev
drwxr-xr-x      29  root  root   3072      Aug    6 09:21    etc
drwxr-xr-x      3   root  root   1024      Jun    18 19:36   home
drwxr-xr-x      4   root  root   3072      Jun    18 19:36   lib
drwxr-xr-x      2   root  root   12288     Jun  18 19:26     lost+found
drwxr-xr-x      4   root  root   1024      Jun    18 19:27   mnt
dr-xr-xr-x      56  root  root   0         Aug    6 05:20    proc
drwxr-x---      9   root  root   1024      Aug    6 10:39    root
drwxr-xr-x      3   root  root   2048      Jun    18 19:39   sbin
drwxrwxrwt      6   root  root   1024      Aug    6 10:15    tmp
drwxr-xr-x      20  root  root   1024      Jun    18 19:33   usr
drwxr-xr-x      15  root  root   1024      Jun    18 19:39 var
[root@linuxi /root]# crontab -r
```

```
[root@linuxl /root]# crrontab -l
no crontab for root
[root@linuxi /root]#
```

첫번째 지령은 창조된 파일 listing 의 내용을 보여 준다. 그다음 listing 을 설치하기 위하여 crontab 지령을 주며 설치된 파일을 보기 위하여 crontab -l 을 준다. 다음으로 파일 listing.out 가 실지로 생성되였다는것을 보여 주는 현재등록부의 내용을 주고 그 파일의 내용을 보기 위하여 파일을 결합시킨다. 그리고 설치된 파일을 crontab -r 로 제거한다. 마지막지령으로 준 crontab -l 은 사용자뿌리에 대하여 설치된 crontab 파일이 없다는것을 보여 준다.

체계관리자들은 시간이 흐름에 따라 많은 시간과 자원을 소비하는 일감들의 일정을 수립하는데서 cron 을 많이 리용한다. 야간에 실행되도록 일정이 세워 진 전형적인 과제는 여벌복사들이다.

아래에 혼합되여 있는 실례는 체계관리자가 어떻게 전면여벌복사를 6 일에 실행하고 증가여벌복사를 다른 날들에 실행하는가를 보여 주고 있다. 이것은 《전면여벌복사지령》과 《증가여벌복사지령》들에 대한 실제적인 지령들을 치환하는 혼합실례이다.

```
$ crontab -l

00 2 * * 6 full backup command
15 12 * * 1-5 incremental backup command
```

첫번째 항목은 전면여벌복사이고 두번째 항목은 증가여벌복사이다. 첫번째 항목에서는 minute 가 00 이고 두번째 항목에서는 minute 가 15 이다. 첫번째 항목에서는 hour 가 2 이고 두번째 항목에서는 hour 가 12 이다. 두 항목들에서 monthday 와 month 는 모두 매 monthday 와 month 를 의미하는 값(*)이다. 첫번째 항목에서는 weekday 가 6 이고 두번째 항목에서는 weekday 가 1-5 이다. 선택적인 username 은 실례에서는 규정되지 않았다. 끝으로 여벌복사지령을 주었다.

minute	hour	monthday	month	weekday	username	command
00	2	all	all	6	n/a	full backup
15	12	all	all	1-5	n/a	incremental

체계관리자들은 일별 또는 주별로 동적파일을 찾는데도 cron 을 리용한다. 핵심부파일은 체계상에서 문제들이 생길 때 디스크에 씌여 지는 기억기의 영상이다. 그것들은 문제에 따라 체계상의 다양한 위치에 씌여 질수 있다. 이 파일들은 때때로 문제의 원천을 식별하는데 리용될수 있으므로 체계관리자들은 그것들을 계속 보존하여야 한다. 아래의 find 지령은 한주일동안에 한번도 리용되지 않은 핵심부파일을 찾기 위하여 한주일에 한번 실행되는데 핵심부파일이름을 뿌리의 홈등록부에 있는 파일에 기입한다.

```
00 2 * * 6 find / -name core -atime 7 > /root/core.files
```

체계관리자는 지난 주에 어떤 핵심부파일이 생겨 났는가를 보기 위하여 월요일에 이 파일을 검사한다. 우의 여벌복사실례에서처럼 이 검사는 매주 수요일 2:00 시에 실행된다.

사용자들은 때때로 체계의 부하가 크지 않을 때 밤새껏 큰 **콤파일**(Compilation)하거나 또는 **일괄처리일감**(Batch Job)을 수행하기 위하여 cron 항목들을 설정한다. 실행되고 있는 cron 일감에 접근하는것을 체계관리자가 거절하지 않는 한 자기의 일감들을 자유롭게 설치할수 있다. 체계관리자는 cron.allow 파일에 있는 cron 을 리용하도록 허락되여 있는 사용자들을 렬거할수 있다. 이 파일에 들어 있지 않는 사용자는 crontab 프로그람을 실행시킬수 없다. cron.allow 가 존재하지 않으면 crontab 파일에로의 접근을 거절당한 사용자들이 있는가를 보기 위하여 cron.deny 를 검사한다.

cron 은 리용하기가 대단히 쉽다. 앞의 실례에서 listing 으로 고찰한것처럼 파일을 창조하고 그 파일에 대하여 crontab 를 실행시키면 된다.

분배하려는 지령을 한번만 실행시키려면 at 지령을 리용할수 있다. at 지령과 지령이 실행되여야 할 시간을 규정해 주고 그다음 "at>"입력재촉문에 실행하려는 지령을 규정해 준다. 아래에서는 /home/martyp 에 있는 모든 핵심부파일들을 오후 9 시(9:00 P.M)에 제거하는 실례를 보여 주고 있다.

```
$ at 9:00PM
at> find /home/martyp -name core exec rm { } \ ;
at>                                         ^d (ctrl+d)를 누른다.
$
```

at 와 시간을 준 후에 지령을 줄것을 요구하는 "at>"입력재촉문이 나타난다. 일반 쉘 재촉문에로 돌아 가려면 다음번 입력재촉문에서 ^**d**(ctrl-d)를 누르면 된다.

망

망에 대하여서는 제 11 장에서 많이 취급되였다. 망은 체계관리자가 설치와 유지에 많은 시간을 소비하는 문제이다. 체계관리자는 체계들이 어떻게 망에 결합되는가를 계획하고 이 계획을 지원하도록 체계들을 조성하는데 많은 시간을 보낸다. 망은 rlogin 으로 원격가입하여 장비된 원격파일체계들을 마치 NFS 체계에 국부적인것처럼 보고 리용하며 많은 중요한 망기능들을 리용하여 다른 체계들에 접근할수 있게 한다. 얼마나 많은 중요한 망지령들이 리용되는가를 알려면 망을 취급한 장을 보면 된다.

syslog 와 일지파일

체계관리자가 체계상의 어떤 문제에 맞다들었을 때 그들은 즉시 체계일지파일을 보기 시작한다. 체계일지파일들은 봉사프로그람들과 핵심부를 포함하는 다양한 원천으로부터 생겨 난다. 전형적인 UNIX 체계상에는 많은 일지파일들이 있으나 여기서는 syslog 라고 부르는 가장 흔히 리용되는 일지파일을 보기로 한다.

대부분의 일지파일들은 /var/log 또는 그 보조등록부들중의 하나, /var/adm 또는 그 보조등록부들중의 하나에 들어 있다. 거의 모든 UNIX 변종들은 /var 안의 그 어디엔가에 일지파일을 넣으므로 적어도 탐색을 시작할 위치를 알고 있다. 일부 일지파일들은 접근하는데 **상급사용자권한**(Super User Right)을 요구하므로 체계상에서만 일지파일을 읽을수 있다.

가장 자주 리용되는 일지파일은 syslog 이다. 이 파일은 포괄적인 가입수단이기때문에 체계사건가입자가 호출한다. 이 파일에는 핵심부, 우편체계, 인쇄기접속기, 크론, 기타 다른 프로그람들이 리용할수 있는 많은 능력들이 포함되여 있다. syslog 는 데몬, 서고루틴, 가입자를 포함하여 여러 부분들로 이루어 져 있다. 체계는 syslogd 라고 부르는 데몬을 실행시키는데 이것을 ps 로 검사할수 있다.

syslog 를 리용한 결과는 체계에서 맞다들수 있는 문제들을 볼수 있게 하는 일지파일이다. syslog 에서 다양한 통보문들을 얻을수 있다.

dmesg

체계정보를 전달하는데 리용되는 다른 수단은 dmesg 이다. dmesg 는 체계완충기에서 최근에 인쇄된 **진단통보**(Diagnostic Message)를 살펴 보고 체계를 시동할 때 생성된 통보문을 인쇄한다.

핵심부

핵심부(Kernel)는 UNIX 체계의 심장부이다. 핵심부의 조성과 유지는 체계관리자가 수행하여야 할 일이다. 이 항목에서는 체계사용자들이 자주 리용할수 있는 핵심부의 중요한 기능들을 서술한다.

일반적으로 UNIX 조작체계는 세개 준위 즉 하드웨어, UNIX 핵심부, 사용자준위프로그람으로 이루어 진다.

UNIX 는 이 책에서는 취급할수 없는 매우 광범하고 다양한 하드웨어상에서 실행되는 다방면적인 조작체계이기때문에 하드웨어에 대하여서는 론의하지 않았다. 이 책에서 취급하는 대부분의 내용들은 사용자준위의것이다. 또한 체계관리작업과 결합된 지령들도 사용자준위의것이다. 핵심부는 모든 사용자준위지령들 및 프로그람들과 하드웨어사이의 대면부를 관리한다.

핵심부가 하드웨어와 사용자준위프로그람들사이의 대면부를 취급하는 방법의 한가지 실례는 파일체계이다. 파일을 읽으려는 요구가 있을 때마다 핵심부는 그 파일을 보려고 접근하는 하드웨어와 그 파일을 읽을것을 요구하는 사용자사이의 대면부를 취급한다.

핵심부에 의하여 제공된 다른 한가지 기능은 사용자가 체계를 독립적으로 리용할수 있게 하는것이다. 대부분의 UNIX 체계들에서는 한 시각에 적은 개수의 프로그람들만이 실행될수 있다. 그러나 동시에 실행되여야 할 프로그람들이 많을수 있다. 핵심부는 주어진 시각에 어느 프로쎄스가 체계 CPU 를 리용하겠는가를 관리하며 한 프로쎄스로부터 다

른 프로쎄스에로 CPU 를 넘기는것을 조종한다. 이 **문맥절환**(Context Switching)은 매 초마다 여러번 진행되며 이것은 핵심부의 가장 복잡한 기능들중의 하나이다.

체계관리자는 특별히 핵심부를 정규적인 기초우에서 갱신한다. 핵심부에는 체계실행을 개선하기 위하여 조절되여야 할 파라메터들이 있다. 특정의 하드웨어를 지원하는 기능들은 핵심부에 포함되여야 한다.

많은 개선된 UNIX 체계들은 체계가 실행되고 있을 때 모듈들을 동적으로 적재한다.

핵심부모듈들을 동적으로 적재하기 위하여 매 모듈에 체계파일을 따로따로 제공해 주어야 한다. 특별히 창조된 모듈들은 체계를 재시동함이 없이 핵심부에 적재되거나 핵심부로부터 제거될수 있다. 이것이 UNIX 변종들에서 찾아볼수 있는 개선된 기능이다.

핵심부의 유지를 책임진 사람은 체계관리자이지만 체계상의 모든 사용자들이 핵심부의 기능을 리용한다.

장치파일

UNIX 체계상의 **장치파일**(Device File)들은 프로그람들이 하드웨어체계와 통신할수 있는 가능성을 마련해 준다. 앞에서는 핵심부가 UNIX 체계상의 하드웨어를 지원하는 방법에 대하여 취급하였다. **장치구동프로그람**(Driver)이 핵심부에 적재되여야 통신하려고 하는 하드웨어가 핵심부에 의하여 조종될수 있다.

일반적으로 말하여 UNIX 체계상의 장치파일들은 **기호장치**(Character Device) 또는 **블로크장치**(Block Device)들이다. 기호장치들은 자료의 입출력완충을 관리하는 UNIX 체계의 개념들과 구동프로그람을 요구한다. 블로크장치들은 파일체계와 그 장치들에 대한 **완충**(Buffering)을 수행할 행을 요구한다. 대부분의 하드디스크구동기들은 블로크 및 기호장치파일들을 둘 다 가지고 있다. 그러므로 하드디스크구동기들은 유연하게 리용된다.

장치파일은 특정의 장치에 대한 중요한 정보를 가지는 UNIX 핵심부를 제공한다. UNIX 핵심부는 입출력조작이 수행되기전에 많은 장치들을 읽어야 한다. 장치파일들은 보통 /dev 등록부에 있다. /dev 안에는 후에 장치파일들을 분류하는데 리용될 보조등록부들이 있을수 있다. 보조등록부의 실례로 디스크장치파일들이 배치될수 있는 /dev/dsk 와 테프구동기장치파일이 배치되여 있는 /dev/rmt 을 들수 있다.

장치파일에 대한 표상을 가지려면 그림 13-4 를 보면 된다. 그림 13-4 는 공통적인 장치파일이름짓기방법을 보여 주는 HP-UX 의 실례이다.

/dev/dsk/c0t1d0 에 대한 디스크종류(description)는 쪼각(slice) 또는 구획, 장치파일의 맨 끝에 나타나는 번호를 가진다. 일부 UNIX 체계들은 디스크를 구역들로 분할하며 다른 체계들은 디스크들이 리용되는 방식을 조종하는데 고준위기록권관리프로그람을 리용한다. 디스크관리는 진행되는 방식에 따라 디스크상의 구역번호들을 볼수도 있고 보지 않을수도 있다.

그림 13−4. 공통적인 장치파일이름짓기방법

아래의 두 실례들은 그림 13−4 에 있는 장치들의 목록들을 보여 준다. 첫번째 목록은 테프구동기의것이고 두번째 목록은 디스크의것이다.

```
$ 1s -1 /dev/rmt/0m
crw-rw-rw- 2 bin bin 205 0x003000 Feb 12 03:00 /dev/rmt/0m
$ ls -1 /dev/dsk/c0t1d0
brw-r----- 1 bin sys 31 0x001000 Feb 12 03:01 /dev/dsk/ c0t1d0
```

장치파일목록에는 쪼각 또는 구획이 없고 /dev/dsk/c0t1d0 목록에는 번호가 있다. 이 목록이 얻어 진 체계는 디스크들이 조성되고 관리되는 방식을 취급하는 기록권관리프로그람들을 리용한다. 디스크들은 이 장치에서는 분할되지 않는다. 그러므로 실례에는 디스크쪼각들이 없다.

UNIX 체계상에는 매 장치파일들을 구성하는 중요한 성분들이 있다. 첫번째는 큰 수이고 두번째는 작은 수이다. 큰 수는 핵심부가 장치를 쓰기 위하여 리용하는 장치구동프로그람의 번호를 가리킨다. 작은 수는 어느 물리적장치가 체계에 련결되였는가를 핵심부와 장치구동프로그람에 알려 준다. 우의 디스크실례에서는 디스크구동프로그람을 위하여 큰 수 31 과 작은 수 0x001000 을 리용한다. 이 수들은 UNIX 변종들에서 크게 차이난다.

테프구동장치파일 /dev/rmt/0m 은 기호장치구동프로그람 stape 에 대응하는 큰 수 205

를 보여 준다. 디스크구동장치파일 /dev/dsk/c0t6d0 은 블로크장치구동프로그람 sdisk 에 대응하는 큰 수 31 을 보여 준다. 테프구동기는 한 기호장치파일만을 요구하기때문에 이것은 테프구동기에 대하여서만 존재하는 장치파일이다. 다른 한편 디스크는 블로크장치 또는 기호장치로 리용될수 있다. 그러므로 기호장치파일 /dev/rdsk/c0t6d0 을 아래의 실례에서 보여 주는바와 같이 큰 수 188 로 고찰한다.

$ ls /dev/rdsk/c0t0d0

crw-r----- 1 root sys 188 0x001000 Feb 12 03:01
/dev/rdsk/c0t1d0

큰 수와 작은 수들은 물론 장치파일들을 이름짓는 방법은 UNIX 변종들에서 차이난다. 그러나 일반적으로 모든 UNIX 변종들상의 장치파일들에 결합된 큰 수와 작은 수들을 볼수 있다. 사용자는 장치파일이름과 번호를 가지고 대화하지 않지만 이러한 내용을 아는것이 좋다.

쏘프트웨어관리

대부분의 UNIX 변종들은 쏘프트웨어를 관리하는 도구를 가지고 있다. 이 도구는 쏘프트웨어로 일하도록 하는 일련의 지령들로 이루어 진다. 쏘프트웨어관리의 기초는 쏘프트웨어이다. 쏘프트웨어는 련관되는 파일들의 그룹이며 지령들은 그 패키지들을 가지고 작업할수 있게 하는 봉사프로그람이다. UNIX 변종들에서 쏘프트웨어들이 개념적으로 어느정도 류사하지만 봉사프로그람들의 이름과 그것들의 조작은 차이난다. 봉사프로그람들은 패키지에 대한 정보를 얻기, 패키지의 첨가와 제거, 패키지의 검사 등 쏘프트웨어와 관련되는 많은 과제들을 수행할수 있다. 조작체계, 응용프로그람들 그리고 다른 쏘프트웨어들이 UNIX 변종상에서 실행되는 쏘프트웨어봉사프로그람들로부터 적재되고 유지되기때문에 체계관리자들은 쏘프트웨어관리에 익숙되여야 한다. 쏘프트웨어관리봉사프로그람들의 이름들은 보통 봉사프로그람의 기능을 잘 알려준다. 아래의 실례들은 Solaris 와 HP-UX 의 일부 쏘프트웨어봉사프로그람들을 보여 준다.

Solaris pkg 지령들:

martyp $ **ls -l /usr/sbin/pkg***

```
-r-xr-xr-x   2  root  sys   102980   Oct  6   1998  /usr/sbin/pkgadd
-r-xr-xr-x   2  root  sys   102980   Oct  6   1998  /usr/sbin/pkgask
-r-xr-xr-x   1  root  sys   157836   Oct  6   1998  /usr/sbin/pkgchk
-r-xr-xr-x   1  root  sys   51084    Oct  6   1998  /usr/sbin/pkgmv
-r-xr-xr-x   1  root  sys   81296    Oct  6   1998  /usr/sbin/pkgrm
```

martyp $ ls -1 /usr/bin/pkg*

```
-r-xr-xr-x   1  bin   sys   97108     Oct  6    1998  /usr/bin/pkginfo
-r-xr-xr-x   1  bin   bin   118348    Oct  6    1998  /usr/bin/pkgmk
-r-xr-xr-x   1  bin   sys   84356     Oct  6    1998  /usr/bin/pkgparam
-r-xr-xr-x   1  bin   bin   29848     Oct  6    1998  /usr/bin/pkgproto
-r-xr-xr-x   1  bin   bin   68112     Oct  6    1998  /usr/bin/pkgtrans
```

HP-UX sw 지령들:

```
martyp $ ls -l /usr/sbin/sw*
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swacl
-r-xr-xr-x   1  bin   bin   446464    Dec  15   1998  /usr/sbin/swagentd
-r-xr--r--   1  bin   bin   16384     Jun  10   1996  /usr/sbin/swapinfo
-r-xr-xr-x   1  bin   bin   24576     Jun  10   1996  /usr/sbin/swapon
-r-xr-xr-x   1  bin   bin   71992     May  30   1996  /usr/sbin/swcluster
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swconfig
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swcopy
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swdepot
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swinstall
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swjob
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swlist
-r-sr-xr-x   2  root  bin   770048    Nov  24   1998  /usr/sbin/swmodify
-r-sr-xr-x   2  root  bin   770048    Nov  24   1998  /usr/sbin/swpackage
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swreg
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swremove
-r-sr-xr-x  11  root  bin   1609728   Dec  15   1998  /usr/sbin/swverify
```

AIX 쏘프트웨어관련의 지령들은 lslpp, installp, lppchk 를 포함한다.

대부분의 UNIX 변종들은 쏘프트웨어관리를 도형대면부를 통하여 수행할수 있게 한다. 이 대면부는 보통 그 체계상에서 실행되는 일차적인 체계관리프로그람으로부터 호출된다. 그림 13-5 는 Red Hat Linux 상에서 rpm 이라고 부르는 도형봉사프로그람의 화면을 보여 준다.

Gnome RPM 창문의 왼쪽 면은 쏘프트웨어의 부류들을 보여 준다. 패키지부류들은 Amusements, Applications, Developments, SystemEnviro-nment, UserInterface 이다. 그림에서는 Development 아래의 Tools 가 선택되여 있다.

그 창문의 오른쪽 면은 도구패키지들을 보여 주는데 현재 make 가 선택되여 있다. 한개 패키지를 선택한 후에 Install, Upgrade, Unselect 등과 같은 다양한 과제들을 수행할수 있다는것을 이 창문의 꼭대기에서 볼수 있다. 만일 Query 를 선택하면 그림 13-6 에서 보여 주는 창문이 나타난다.

그림 13-5. Red Hat Linux Gnome RPM 창문

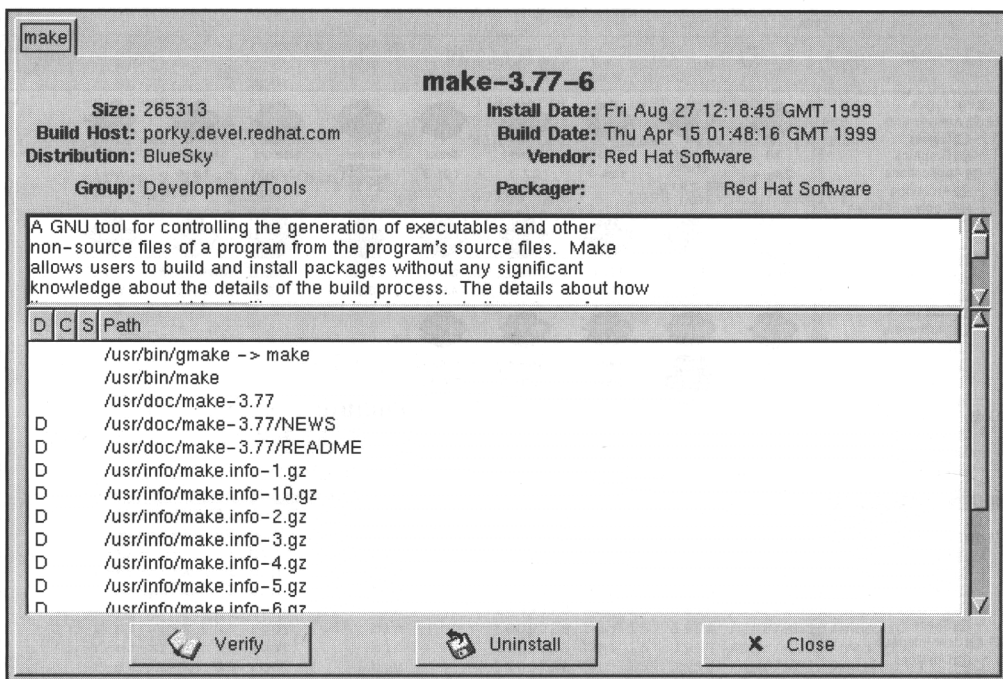



그림 13-6. Red Hat Linux *Package Info* 창문

그림 13−6 은 쏘프트웨어의 교정, 그 크기 등 선택된 make 패키지에 대한 많은 정보들을 보여 주고 있다.

지령행에서 rpm 을 리용하면 거의 동일한 정보를 얻을수 있다. 아래의 실례는 make 에 대한 정보를 생성하도록 선택항목을 붙여 rpm 지령을 준데 대하여 보여 준다.

[root@linux1 /root]# **rpm - -query -qiv make**

```
Name          : make                     Relocations:  /usr
Version       : 3.77                          Vendor:  Red Hat Software
Release       : 6                         Build Date:  Thu Apr 15 09:48s:16 1999
Install date: : Fri Aug 27 08:18:45 1999  Build Host:  porky.devel.redhat.com
Group         : Development/Tools         Source RPM:  make-3.77-6.src.rpm
Size          : 265313                       License:  GPL
Packager      : Red Hat Software<http : //developer.redhat.com/bugzilla>
Summary       : A GNU tool which simplifies the build process for users.
Description   :
A GNU tool for controlling the generation of executables and other
non-source files of a program from the program's source files. Make
allows users to build and install packages without any significant
knowledge about the details of the build process. The details about how
the program should be built are provided for make in the program's
makefile.

The GNU make tool should be installed on your system because it is
commonly used to simplify the process of installing programs.
```

일부 체계관리자들은 rpm 과 같은 도구들의 도형적방안인 지령행 vs 로 작업하는것을 더 좋아 한다. 대부분의 다른 UNIX 변종들은 앞에서 본바와 같이 쏘프트웨어관리도구들뿐아니라 일반 관리도구들에 대한 도형대면부를 포함하고 있다. 여러가지 다양한 방식으로 UNIX 체계상에서 동일한 결과를 얻을수 있으며 특별히 체계관리에 관해서는 좋아 하는 방법을 리용하면 된다.

인 쇄

UNIX 체계들은 인쇄와 관련되는 다양한 프로그람들을 실행시킨다. 이 프로그람들은 본문파일들, 형식화된 문서들, 도형파일들 등의 인쇄를 지원한다. 이 항목에서는 인쇄기관리와 인쇄에 대한 기초개념들을 취급한다.

mbox 라고 부르는 파일을 인쇄하려면 아래의 실례에서 보여 주는바와 같이 lp 지령을

리용하면 된다.

```
martyp $ lp mbox
Job number is: 1
martyp $
```

이 실례에서는 파일 mbox 가 인쇄될것을 요구하였다. lp 로부터 요구식별번호를 가리키는 통보문을 받는다.

다음의 실례에서 보여 주는바와 같이 한개 lp 지령으로 여러개 파일들을 인쇄할수 있다.

```
martyp $ ls -l
total 80
-rw-------    1  martyp  usr     350  Sep  27  07:22  dead.letter
-rw-rw-r--    1  martyp  usr   24576  Sep  6   07:07  inst.out
-rw-rw-r--    1  martyp  usr       0  Aug  21  06:45  lanadmin.list
-rw-------    1  martyp  usr    1485  Sep  27  08:20  mbox
-rw-rw-r--    1  martyp  usr     353  Sep  29  04:57  trip
-rw-rw-r--    1  martyp  perf    635  Mar  21  1999   typescript
martyp $ lp t*
Job number is: 2
martyp $
```

이 실례에서는 "t"로 시작되는 두 파일이 인쇄되였으며 한개 일감번호가 두 파일의 인쇄와 결합되였다.

많은 UNIX 체계들에는 여러개 인쇄기들이 련결되여 있다. 인쇄기에로 파일을 전송하려면 선택항목 **-d** 뒤에 인쇄기이름을 붙여 주면 된다. 앞의 실례에서는 기정체계인쇄기를 리용하였다.

mbox 의 인쇄를 수행할 인쇄기장치를 아래의 실례에서 보여 주는것처럼 선택항목 **-d** 로 규정해 줄수 있다.

```
martyp $ lp -d ros2228 mbox
Job number is: 3
martyp $
```

이 출력은 파일 test 를 규정된 인쇄기에로 전송하며 다시 일감번호가 규정된다. 아래의 실례에서 보여 주는바와 같이 인쇄기이름을 규정해 주지 않았을 때에는 리용될 기정인쇄기를 규정할수 있다.

```
martyp $ LPDEST=ros2228
martyp $ export LPDEST
martyp $ lp mbox
Job number is: 4
martyp $
```

환경변수 **LPDEST** 는 사용자의 기정인쇄기와 결합된다. 기정인쇄기를 시동파일에서 규정해 줄수 있다.

인쇄일감들은 인쇄대기렬에 들어 서므로 사용자는 다른 작업을 진행할수 있다. 즉 인쇄일감이 완성될 때까지 기다리지 않아도 된다. 아래의 실례에서 보여 주는바와 같이 선택항목 -m 을 리용하면 인쇄할 때 문제가 생기는 경우에 **전자우편통보문**(electronic mail message)을 받을수 있다.

```
martyp $ lp -m t*
Job number is: 5
martyp $
```

여기서는 "t"로 시작되는 두 파일들을 앞서 설치한 기정인쇄기에 다시 보낸다. 다른 것을 알려 주는 전자우편통보문을 받지 않는 한 이 인쇄일감은 아무런 문제없이 완성될 수 있다.

때때로 인쇄기의 상태를 보려고 할수 있다. 인쇄대기렬조성기능이라는것은 여러 파일들이 인쇄대기렬에 들어 선다는것을 의미하는데 이것은 사용자가 파일을 인쇄하려면 기다려야 한다는것을 말한다. 인쇄대기렬에서 사용자의 파일앞에 한개 파일만이 있을수 있다. 그러나 그것은 ERP 체계로부터의 보고서와 같은 대단히 큰 파일일수 있다. 인쇄기들의 상태를 얻으려면 아래의 실례가 보여 주는바와 같이 lpstat 지령에 선택항목 -t 를 붙여 주면 된다.

```
martyp $ lpstat -t
Queue    Dev     Status   Job     Name          From            To
                 Submitted      Rnk  Pri  Blks     Cp      PP%
------------------------------------------------------------------------------------------------
a464     a464d  READY
a464:
    no entries
a438     a438d  READY
a438:
    no entries
a570     a570d  READY
a570:
    no entries
```

```
a654       a654d  READY
a654:
      no  entries
a662       a662d  READY
a662:

a662:

a662:
a662:

a662:

a946       a946d  READY
a946:
      no  entries
a956       a956d  READY
a956:
      no  entries
a732       a732d  READY
a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down

a732:  ros-ps4:  Warning:  a732  is  down
```

```
ros2227      ros22  READY
ros2227:
         no  entries

ros2228      ros22  READY
ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

ros2228:
ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

ros2228:

martyp  $
```

이 출력은 모든 인쇄기들의 상태를 보여 준다. 이를 통하여 체계에 많은 인쇄기들이 련결되여 있다는것을 알수 있다. 책에 들어 있는 내용이 너무 길어 지기때문에 실례에서는 여러개 인쇄기들을 생략하였다. 어떤 인쇄기들은 인쇄할수 없다는것을 의미하는 경고통보문을 내보낸다. 이런 문제들을 처리하는 방법에 대하여서는 여기에서 취급하지 않는다. 왜냐하면 이것은 체계관리자가 독립적으로 취급해야 할 문제이기때문이다. 설치된 인쇄기들의 상태를 보려면 아래의 실례가 보여 주는바와 같이 선택항목 -d 을 주면 된다.

```
martyp  $ lpstat  -d
Queue    Dev    Status  Job  Files    User    PP  %    Blks  Cp  Rnk
------------------------------------------------------------------------------------------------
ros2228  ros22      READY
ros2228:
ros2228:
```

ros2228:
ros2228:
ros2228:
ros2228:
martyp $

일감이 제기되였을 때 그것을 cancel 지령으로 제거할수 있다. 앞의 실례들에서 본 일감번호를 보고 인쇄기이름을 cancel 지령과 함께 리용할수 있다. 보통 작은 인쇄일감은 취소하려고 하면 빨리 처리되고 인쇄될것이지만 큰 인쇄일감은 완결되기전에 취소될수 있다.

lpstat 지령은 일감을 제기하였을 때 일감번호를 적어 두지 않은 경우에 그 일감번호를 얻어 내는데 리용될수 있다.

표 13-2 는 흔히 리용되는 일부 lp 관련의 지령들을 보여 준다. 그것들중 일부는 인쇄기조성과 같은 체계관리작업과 결합되여 있으며 사용자들은 보통 리용할수 없다.

표 13-2 lp 지령들

지 령	설 명
/usr/sbin/accept	일감들을 대기렬에 받아 들이는 접수를 시작한다.
/usr/bin/cancel	대기렬에 들어 선 인쇄일감들을 취소한다.
/usr/bin/disable	장치로 인쇄할수 없다.
/usr/bin/enable	장치로 인쇄할수 없다.
/usr/sbin/lpfence	인쇄하여야 할 파일에 최소우선권을 설정 (모든 UNIX 변종들에서 리용할수 있는것이 아니다.)
/usr/bin/lp	일감들을 인쇄하기 위하여 대기렬에 접수한다.
/usr/sbin/lpadmin	제공된 선택항목들을 가지고 인쇄체계를 조합한다.
/usr/sbin/lpmove	인쇄일감들을 한 장치에서 다른 장치에로 옮긴다.
/usr/sbin/lpsched	lp 일정작성데몬을 시동한다.
/usr/sbin/lpshut	lp 일정작성데몬을 정지한다.
/usr/bin/lpstat	제공된 선택항목들에 기초하여 인쇄상태를 보여 준다.
/usr/sbin/reject	일감들을 대기렬에 받아 들이는 접수를 정지한다.

도형에 기초한 관리도구

대부분의 UNIX 변종들은 도형에 기초한 체계관리도구들을 갖추고 있다. 보다 더 명백히 말하면 그것들은 기호 및 도형대면부를 가진 차림표구동형도구들이다. 도형형식으로만 되여 있는것도 일부 있다. 이 항목에서 취급되는 문제들을 포함하여 체계관리기능들은 거의 도형적인 도구들을 가지고 수행될수 있다. 그러한 도구들에 대한 표준은 없으므로 그것들은 UNIX 변종들마다 차이난다.

일부 체계관리자들은 자기들의 작업을 지령행에서 수행하며 다른 체계관리자들은

기초적인 과제들을 도형적인 도구로 수행한다. 또 다른 체계관리자들은 전체 또는 일부 체계관리과제들을 도형적인 도구로 수행한다. 그 방법은 체계관리자의 선택에 의존한다.

다음의 그림들(그림 13-7, 그림 13-8, 그림 13-9, 그림 13-10, 그림 13-11, 그림 13-12)은 각각 Solaris, HP-UX, AIX, Red Hat Linux 상의 도형적인 체계관리도구들을 리용할 때 볼수 있는 **고준위대면부**(Top-Level Interface)를 보여 준다.

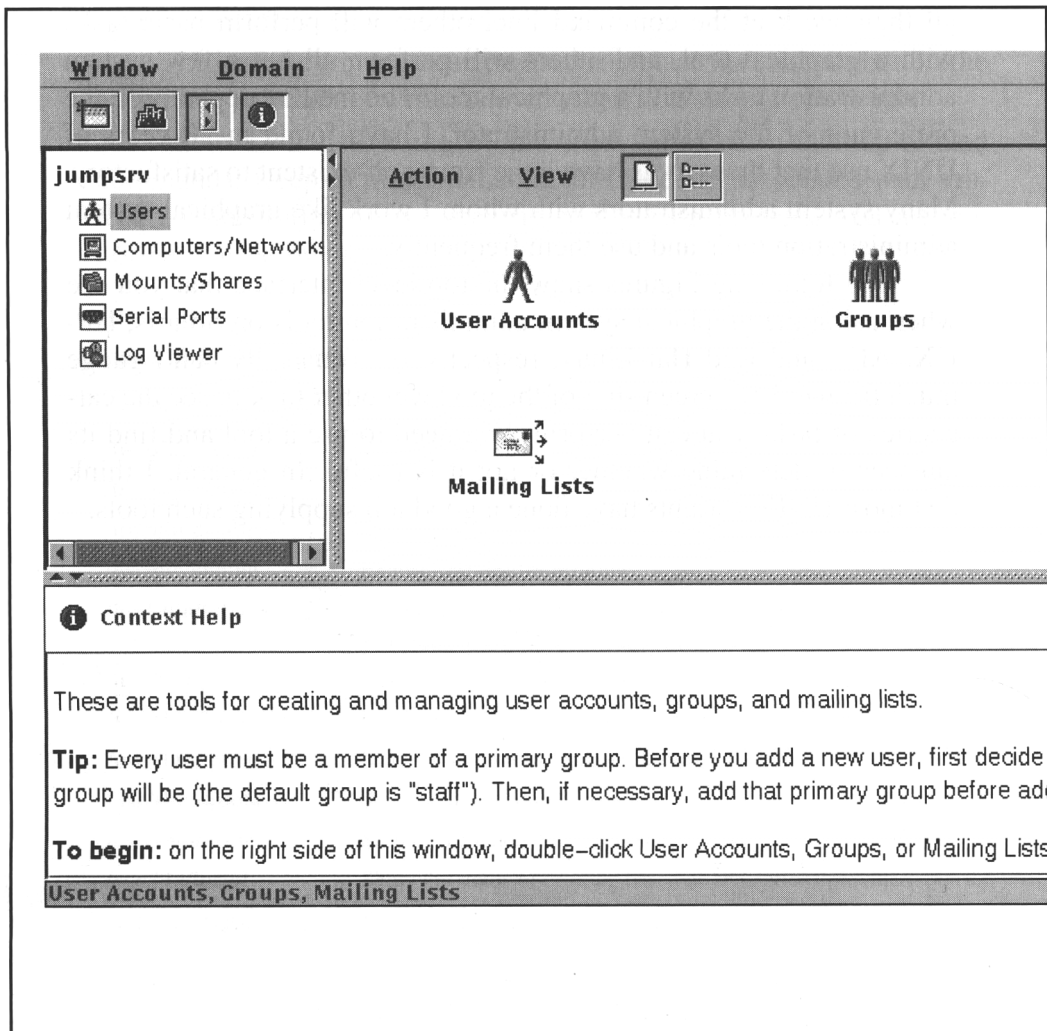



그림 13-7. Solaris 도형체계관리도구-users는 맨 우에, software는 맨 아래. Browse부류들은 다음과 같다. Users, Groups, Hosts, Printers, Serial Ports, and Software

그림 13-8. HP-UX 도형체계관리도구

Exit Show

Return To:

System Management

Software Installation and Maintenance
Software License Management
Devices
System Storage Management (Physical & Logical Storage)
Security & Users
Communications Applications and Services
Print Spooling
Problem Determination
Performance & Resource Scheduling
System Environments
Processes & Subsystems
Applications
Using SMIT (information only)

Cancel

그림 13-9. AIX 도형체계관리도구

그림 13-10. Red Hat Linux 도형체계관리도구-1

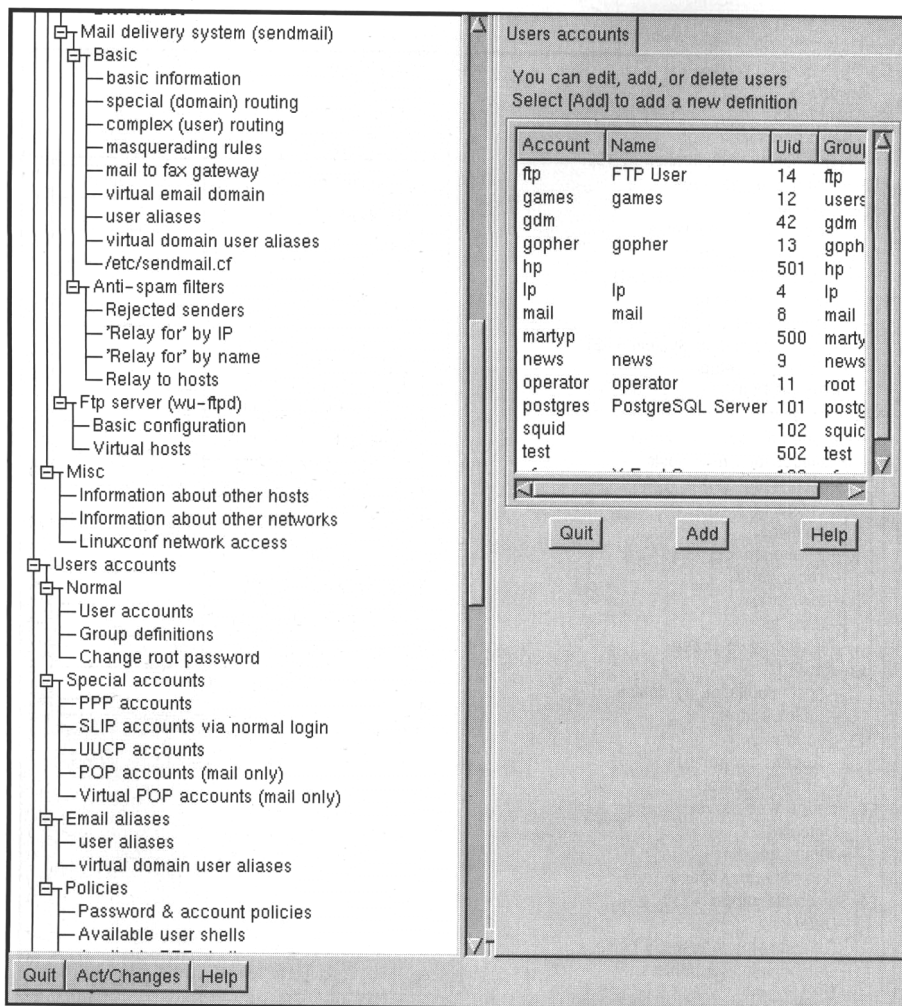



그림 13-11. Red Hat Linux 도형체계관리도구-2

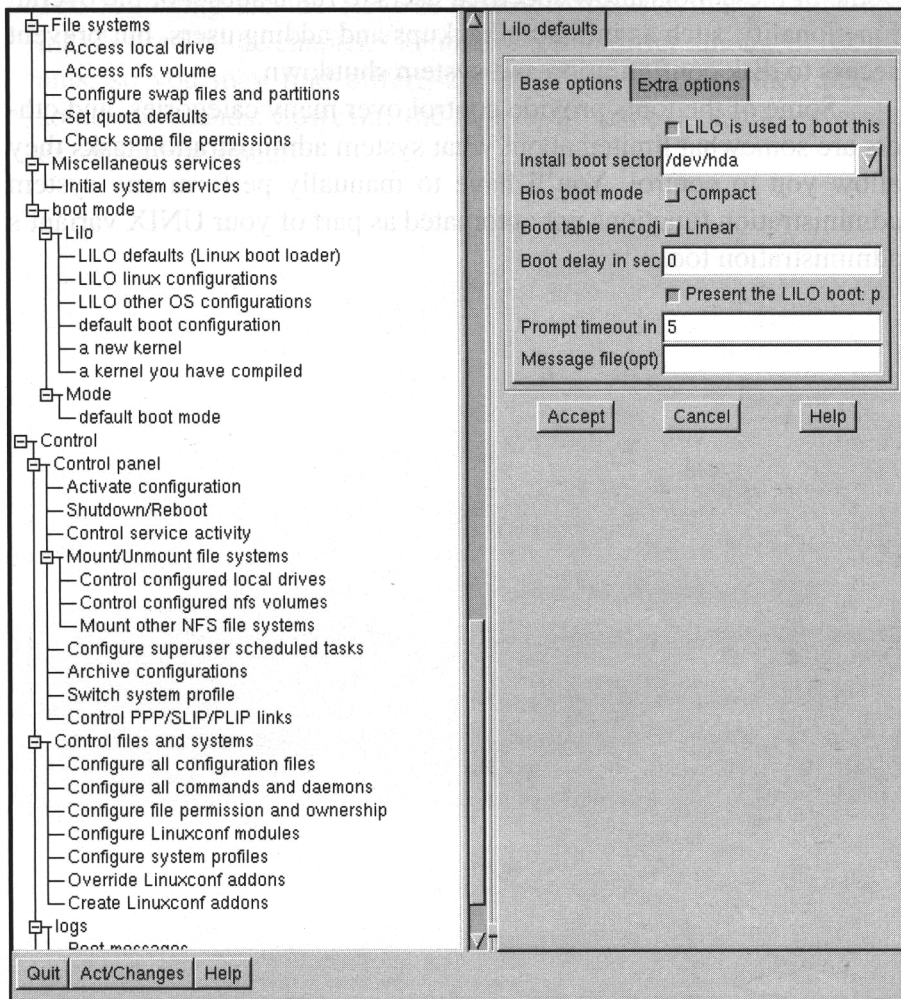



그림 13-12. Red Hat Linux 도형체계관리도구-3

사용자들은 이러한 도구를 리용할수 없다. 체계관리도구를 실행시키려면 특수한 허락이 필요된다. 사용자들이 여벌복사를 초기화하고 사용자들을 첨가하는것과 같은 일부 기능들을 이 도구들로 실행할수 있도록 하자면 디스크조성과 체계끄기에로의 접근을 금지시켜야 한다.

일부 도구들은 많은 과제들을 조종하며 다른 도구들은 어떤 체계관리과제들을 조종할수 있는가에 관하여 어느정도 제한되여 있다. 어떤 UNIX 변종들에서 일부 자동화되지 않은 관리도구의 체계관리지령들은 수동적으로 수행되여야 한다.

지령소개

여기서는 이 장에서 리용된 여러 지령들을 묶어서 보여 주고 있다. 지령들은 UNIX 변종들에서 차이나므로 일부 지령들에 대한 선택항목 또는 다른 령역에서 무엇이 차이나는가를 알아 보아야 한다. 아래의 지령들은 좋은 기준으로 리용될수 있다.

cron

cron – 지령들을 특정한 날자 혹은 시간에 실행시킨다.

---

cron(1M)

이름

cron – 일감을 정해진 시간에 실행시키는 데몬

형식

/usr/sbin/cron

해설

cron 은 지령들을 규정된 날자와 시간에 실행시킨다. 정기적으로 예정된 지령들은 crontab 파일에 배치된 지령들에 의하여 규정된다. 사용자들은 자기의 crontab 파일들을 하나의 crontab 지령으로 제출한다(crontab(1)을 참고). 사용자들은 실행되여야 할 지령들을 at 또는 batch 지령으로 한번만 실행시킨다.

cron 이 전혀 존재하지 않으면 그 지령들은 한번만 실행되게 된다. cron 을 시동스크립트 /sbin/init.d/cron 으로 초기화프로쎄스에서 실행시키는것이 가장 좋다(init(1M)을 참고).

cron 은 프로쎄스를 초기화하면서 그리고 어느한 파일이 첨가되거나 삭제되거나 갱신되였다는것이 at, batch 혹은 crontab 에 의하여 통보될 때 crontab 파일들과 at/batch 지령파일들에 대한 일정을 세운다.

cron 이 어느한 일감을 실행시킬 때 그 일감의 사용자 ID 와 그룹 ID 는 그 일감을 제출한 사용자의것으로 설정된다.

봄과 가을의 시간변환

동틀무렵절약시간(Daylight Savings Time)변환이 진행되는 날들에 (동틀무렵절약시간이 적용되는 **시간대**(Time Zone)와 나라들에서) cron 은 보통과는 차이나게 지령들의 일정을 세운다.

아래의 서술에서 일의적이지 않은 시간은 동틀무렵절약시간변환때문에 동일한 날에 두번 나타나는 시와 분을 가리킨다(보통 가을철에). 존재하지 않는 시간은 동틀무렵절약시간변환때문에 나타나지 않는 시와 분을 가리킨다(보통 여름철에). DST-변위는 표준시간에 적용되여 동틀무렵절약시간을 결과로 주는 변위를 가리킨다. 이것은 보통 한시간이지만 23 시간과 59 분까지의 시간과 분의 결합으로 될수 있다(tztab(4)를 참고).

어느한 지령이 모호한 시간에 실행되도록 규정되였다면 그 지령은 모호한 시간의 첫 출현에서 한번만 실행된다.

어느한 지령이 존재하지 않는 시간에 실행되도록 규정되였다면 그 지령은 DST-변위에 의하여 규정된 시간이 지난 후에 실행된다. 그러한 조절이 그 지령을 실행하도록 규정한 다른 시간과 충돌하게 될 때 그 지령은 동일한 시간에 두번 실행되는것이 아니라 한번만 실행된다.

하루종일 실행되도록 일정을 세운 지령들은(crontab 항목의 시간마당에 *이 있다.) 다른 조절이 없이 일정이 세워 진다.

외부적영향

환경변수들

LANG 은 통보문들이 현시될 언어를 결정한다. LANG 이 규정되지 않았거나 빈 기호렬로 설정되여 있으면 "c"를 기정값으로 본다(lang(5)를 참고). 국제화변수가 무효한 설정을 포함하면 모든 국제화변수들은 "c"를 기정값으로 한다. (eviron(5)를 참고.)

진단

cron 에 의하여 실행되는 모든 지령들의 리력은 /var/adm/cron/log 에 기록되여 있다.

실례

아래의 실례들은 시간대가 MST7MDT 이라고 가정한다. 이 시간대에서 DST 변환은 1 초전 2:00 a.m.에 일어 나며 DST-변위는 1 시간이다. crontab 파일에 있는 다음과 같은 항목들을 고찰하자.

```
# 분  시     월     일     월     요일지령
# -------------------------------------------------------------
  0  01     *      *      *      Job_1
  0  02     *      *      *      Job_2
  0  03     *      *      *      Job_3
  0  04     *      *      *      Job_4
```

0	*	*	*	*	Job_hourly
0	2, 3, 4*		*	*	Multiple_1
0	2, 4	*	*	*	Multiple_2

DST 변환이 진행되는 날의 1:00 a.m.- 4:00 a.m.주기에 대하여 그 결과는 다음과 같다.

일감	가을에 실행된 시간		봄에 실행된 시간	
Job_1	01:00	MDT	01:00	MST
Job_2	02:00	MDT	03:00	MDT
Job_3	03:00	MST	03:00	MDT
Job_4	04:00	MST	04:00	MDT
Job_hourly	01:00	MDT	01:00	MST
	02:00	MDT		
	02:00	MST		
	03:00	MST	03:00	MDT
	04:00	MST	04:00	MDT
Multiple_1	02:00	MDT		
	03:00	MST	03:00	MDT
	04:00	MST	04:00	MDT
Multiple_2	02:00	MDT	03:00	MDT
	04:00	MST	04:00	MDT

경고

동틀무렵절약시간때문에 봄에 존재하지 않는 시간이 있으면 존재하지 않는 시간에 실행되도록 일정이 세워 진 지령은 한번만 실행된다. 실례로 MST7MDT 시간대에서 2:00 a.m.과 2:30 a.m.에 실행되도록 일정이 세워 진 지령은 3:00 a.m.에만 실행된다. 2:30 a.m.에 일정이 세워 진 지령은 전혀 실행되지 않는다.

종속성

HP 프로쎄스자원관리프로그람

선택적인 **HP 프로쎄스자원관리**(Process Resource Management)쏘프트웨어가 설치되고 조성되였으면 일감들은 그 일감의 일정을 세운 사용자의 **초기프로쎄스자원그룹**(Initial Process Resource Group)에 배속된다. 사용자의 초기그룹은 일감의 일정을 세울 때가 아니라 일감이 실행될 때 결정된다. 사용자의 초기그룹이 정의되면 일감은 사용자기정그룹(PRMID=1)에서 실행된다. HP PRM 의 조성을 보려면 prmconfig(1)를, 사용자의 초기프로쎄스자원그룹이 어떻게 결정되였는가를 알려면 prmconf(4)를 보면 된다.

저자

cron은 AT&T와 HP에 의하여 개발되였다.

파일들

/var/adm/cron	기본조성등록부
/var/spool/cron/atjobs	at와 batch일감파일들을 포함하는 등록부
/var/spool/cron/crontabs	crontab파일을 포함하는 등록부
/var/adm/cron/log	가입급수정보

관련항목

at(1), crontab(1), sh(1), init(1M), cdf(4), queuedefs(4), tztab(4)

HP 프로쎄스자원관리프로그람
HP 프로쎄스자원관리프로그람사용자지도서에 있는 prmconfig(1), prmconf (4)

표준일치

cron: SVID2, SVID3

df

df － 빈디스크블로크들의 량을 알려 준다.

---

df(1M)

이름

df(generic) － 빈 파일체계디스크블로크들의 개수를 알려 준다.

형식

/usr/bin/df [-F Fstype] [-befgiklnv] [-t | -P] [-o specific_options] [-V] [special | directory]

해설

df 지령은 **초블로크**(superblock)를 검열하여 파일체계에서 리용할수 있는 빈 513-byte 블로크들의 개수와 빈 마디점들의 개수를 현시한다. 등록부가 규정되지 않았다면 장비된 모든 파일체계상의 빈공간이 현시된다. df에 주는 인수들이 경로이름들이면 df는 그 이름이 붙은 파일을 포함하는 파일체계상의 빈공간을 알려준다. df에 준 인수가 장비되지 않은 파일체계이면 장비되지 않은 그 파일체계에 있는 빈 공간이 현시된다.

선택항목들

df 는 다음과 같은 선택항목들을 가진다.

-b 빈 키로바이트수를 알려 준다.

-e 빈 파일의 개수를 알려 준다.

-f 빈 목록안에 있는 블로크의 개수를 알려 준다.

-F FStype FStype 파일체계형만을 알려 준다(fstype(1M)을 참고).

-g statvfs(2)에서 서술된 전체 구조를 알려 준다.

-i 마디점들의 총 개수, 빈 마디점들의 개수, 리용된 마디점들의 개수, 리용되는 마디점들의 퍼센트를 알려 준다.

-k 할당된 키로바이트수를 알려 준다.

-l 국부적파일 체계에 대하여 알려 준다.

-n 파일체계이름을 알려 준다. 다른 선택항목이 없이 리용되면 장비된 파일체계형들의 목록을 현시한다.

-o specific_options
매 파일체계형에 고유한 선택항목들을 규정한다. specific_options 는 이 지령의 특정한 FStype 모듈을 위하여 마련된 부분선택항목들이 반점으로 분리된 목록이다. 보다 더 상세한것을 알려면 파일체계특성에 대한 안내서를 참고.

-P 파일체계의 이름, 파일체계의 크기, 리용된 블로크들의 개수, 빈 블로크들의 개수, 리용된 블로크들의 퍼센트, 파일체계계층에 따르는 등록부를 알려 준다.

-t 총 할당된 블로크도형과 빈 블로크들의 개수를 알려 준다.

-v 리용된 블로크들의 퍼센트, 리용된 블로크들의 개수, 빈 블로크들의 개수를 알려 준다. 이 선택항목은 다른 선택항목들과 함께 리용될수 없다.

-V 완성된 지령행을 현시할뿐 다른 작용은 수행하지 않는다. 지령행은 사용자가 규정한 선택항목들과 /etc/fstab 으로부터 유도된 다른 정보들을 종합하여 발생된다. 이 선택항목은 사용자가 그 지령행을 확증할수 있게 한다.

외부적영향

환경변수

LC_MESSAGES 는 통보문들이 현시될 언어를 결정한다.

LC_MESSAGES 가 그 환경에서 규정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 의 값이 기정으로 리용된다. LANG 이 규정되지 않았거나 빈 기호렬로 설정되여 있으면 기정값 "C"가 LANG 대신에 리용된다(lang(5)를 참고).

임의의 국제화변수가 무효한 설정값을 포함하면 df 는 모든 국제화변수들이 "C"로 설정된것처럼 동작한다(environ(5)를 참고).

국제적인 코드모임보장
단일바이트와 여러 바이트기호코드모임들이 지원된다.

실례

장비된 모든 파일체계들에 대하여 빈 디스크블로크들의 개수를 알아 본다.

df

장비된 모든 HFS 파일체계들에 대하여 빈 디스크블로크들의 개수를 알아 본다.

df -F hfs

장비된 모든 NFS 파일체계들에 대하여 빈 파일들의 개수를 알아 본다.

df -F nfs -e

장비된 모든 파일체계들에 대하여 할당된 총 블로크도형과 빈 블로크들의 개수를 알아 본다.

df -t

/usr 로서 장비된 파일체계들에 대하여 할당된 총 블로크도형과 빈 블로크들의 개수를 알아 본다.

df -t /usr

파일들

/dev/dsk/* 파일체계장치들
/etc/fstab 파일체계들에 대한 정적인 정보
/etc/mnttab 장비된 파일체계표

관련항목

du(1), df_Fstype(1M), fsck(1M), fstab(4), fstyp(1M), statvfs(2), mnttab(4)

표준일치

df:SVID2, SVID3, XPG2, XPG3, XPG4

du

du – 빈 디스크블로크량을 알려 준다.

---

du(1)

이름

du – 디스크리용정형을 개괄한다.

형식

du [-a | -s] [-brx] [-t type] [name…]

해설

du 지령은 name 에 의하여 규정된 매 등록부와 파일에 할당된 513−바이트블로크들의 개수를 준다. 그 블로크들의 개수는 파일의 간접블로크들을 포괄한다. 둘 또는 그이상의 련결을 가진 파일은 한번만 계수된다. name 이 생략되면 현재작업등록부가 리용된다.

기정으로 du 는 name 인수와 그 계층안에 포함된 매 등록부에 대하여서만 한개 항목을 발생시킨다.

선택항목

du 지령은 다음과 같은 선택항목들을 가진다.

-a	표준적인 출력과 등록부계층에 들어 있는 매 파일들에 대한 항목들을 인쇄한다.
-b	파일체계교환이 가능하게 되는 등록부인 매 name 인수에 대하여 교환체계가 현재 리용하고 있는 블로크들의 개수를 인쇄한다.
-r	읽을수 없는 등록부, 접근할수 없는 파일들에 대한 통보문을 인쇄한다. 표준적으로 du 는 그러한 조건들에 대하여 응답을 하지 않는다.
-s	규정된 매 name 인수에 대하여 총 디스크리용정형을 인쇄한다.
-x	name 인수에 의하여 규정된 파일과 같은 동일한 장치를 가지는 파일들에로의 보고를 제한한다. 디스크리용은 표준적으로 주어진 매 name 인수아래의 전체 등록부계층에 대하여 보고된다.
-t type	규정된 형의 파일체계들에로의 보고를 제한한다.（type 에 대한 실례값들은 hfs, cdfs, nfs 등이다.）여러개의 -t type 선택항목이 규정될수 있다. 디스크리용정형은 표준적으로 주어진 매 name 인수아래의 전체 등록부계층에 대하여 보고된다.

실례

현재 작업등록부와 그 아래의 모든 등록부들에 대한 디스크의 리용정형을 현시한다. 읽을수 없는 등록부들에 대하여서는 오유통보문을 현시한다.

du -r

장비된 cdf 또는 nfs 파일체계들을 제외하고 전체 파일체계들에 대한 디스크리용정형을 현시한다.

du -t hfs /

뿌리기록권상의 파일체계들에 대하여서만 디스크리용정형을 현시한다. 장비된 다른 파일체계들에 대하여서는 리용통계가 종합되지 않는다.

du -x /

경고

구멍을 포함하는 파일체계들에 대해서는 블로크들의 개수가 정확하지 못하다.

관련항목

df(1M), bdf(1M), quot(1M)

표준일치

du:SVID2, SVID3, XPG2, XPG3, XPG4

group

group – 그룹들에 대한 정보를 포함하는 파일

---

group(4)

이름

group, logingroup – 그룹파일, grp.h

해설

group는 매 그룹에 대하여 다음의 정보를 포함한다.

- 그룹이름
- 코드화된 통과암호
- 수값그룹 ID
- 그룹에서 허락될 모든 사용자들이 반점으로 분리된 목록

이것은 ASCII 파일이다. 마당들은 두점으로 분리되여 있으며 매 그룹은 행바꾸기로 분리된다. 공백은 마당들을 분리하는데 리용될수 없다. 통과암호마당이 null이면 그룹에 결합된 통과암호는 없다.

체계에는 이러한 형식을 가진 두개의 파일 즉 /etc/group 과 /etc/logingroup 이 있다. 파일 /etc/group 는 매 그룹의 이름을 주며 newgrp 봉사프로그람에 의한 그룹변경을 지원하기 위하여 존재한다(newgrp(1)을 참고). /etc/logingroup 은 login 과 initgroup()를 통하여 매 사용자에 대한 기정그룹접근을 제공한다(login(1)과 initgroups(3c)를 참고).
매 사용자에 대하여 설치된 실제적이고 효과적인 그룹 ID 는 /etc/passwd 에 정의되여 있다(passwd(4)를 참고). /etc/logingroup 이 비였거나 존재하지 않으면 기정그룹접근목록은 비게 된다. /etc/logingroup 과 /etc/group 이 동일한 파일에로의 련결이면 기정접근목록은 사용자와 결합된 전체 그룹모임을 포함한다. /etc/logingroup 에 있는 그룹이름과 통과암호마당들은 전혀 리용되지 않는다. 즉 그것들은 단지 그 두개 파일들을 단일한 형식으로 주기 위하여 도입된것이다.
/etc/logingroup 또는 /etc/passwd 에서 리용되는 모든 그룹 ID 들은 /etc/group 에서 정의되여야 한다. /etc/logingroup 에 있는 NGROUPS 개 그룹이상에 결합된 사용자는 있을수 없다(setgroups(2)를 참고).

이 파일들은 등록부 /etc 에 갖추어 져 있다. 코드화된 통과암호때문에 그 파일들은 일반적인 쓰기허락을 가질수 없으며 수값그룹 ID 들을 이름들에 대응시키는데 리용될수 있다.

그룹구조는 <grp.h>에 정의되여 있으며 다음과 같은 성원들을 포함한다.

```
char  *gr_name;    /*그룹의 이름*/
char  *gr_passwd; /*코드화된 그룹통과암호*/
gid_t  gr_gid;                 /*수값그룹 ID*/
char  **gr_mem;   /*성원이름들을 가리키는 지적자들의 배렬*/
```

망특성

NIS

파일 /etc/group 은 망정보봉사(NIS)로부터의 항목들을 합쳐 놓을것을 의미하는 기호 +로 시작되는 행을 포함할수 있다. 두가지 형태의 +항목들이 있다. 즉 +는 그 위치에 NIS 그룹파일의 전체 내용들을 삽입할것을 의미하며 +name 은 그 위치에 NIS 로부터의 name 에 대한 항목을 삽입할것을 의미한다. +항목이 null 이 아닌 통과암호 또는 그룹성원마당을 가지면 그 마당의 내용들은 NIS 에 포함된것을 무시한다. 수값그룹 ID 마당은 무시될수 없다.

그룹파일은 -로 시작되는 행도 가질수 있다. 이 항목들은 그룹항목들을 거부하기 위하여 리용된다. -항목의 형태는 한가지이다. 즉 -name 으로 이루어 지는 항목은 name 에 대한 다음번 항목을 거부할것을 의미한다. 그 항목들은 다음번 항목이 NIS 로부터 오는가 아니면 국부적인 그룹파일로부터 오는가를 고려함이 없

이 거부된다.

경고

그룹파일들은 빈 행을 포함할수 없다. 빈 행은 그 파일을 리용하는 체계관리쏘프트웨어에서 예측할수 없는 현상을 일으킬수 있다.

그룹 ID(gid) 9 는 파스칼언어조작체계와 BASIC 언어조작체계를 위하여 예약되여있다. 이것들은 HP-UX 와 동일한 디스크에 공존할수 없는 300/400 계렬콤퓨터들을 위한 조작체계들이다. 이 gid 를 다른 목적에 리용하면 파일전송과 공유를 진행할수 없다.

/etc/group 에 있는 매 행의 길이는 <limits.h>에서 정의된대로 LINE_MAX 로 제한되여 있다. 매 그룹당 최대 사용자수는 (LINE_MAX-50)/9 이다.

/etc/group 이 /etc/logingroup 에 련결되여 있으면 사용자의 그룹소속은 NIS 에 의하여 관리된다. 그리고 응답할수 있는 NIS 봉사기가 없으므로 사용자는 어느 한 봉사기가 응답할 때까지 체계에 가입할수 없다.

종속성

NIS

실례

파일 /etc/group 의 본보기는 다음과 같다.

```
other:*:l:root, daemon, uucp, who, date, sync
-oldproj
bin:*:2:root, bin, daemon, lp
+myproject: : :bill, stave
+ :
```

그룹 other 의 gid 는 1 이며 성원들은 root, daemon, uucp, who, date, sync 이다. 그룹 oldproj 는 선택항목 -oldproj 다음에 나타나기때문에 무시된다. 또한 그룹 myproject 는 성원 bill 과 steve 그리고 그 그룹에 대한 NIS 항목의 통과암호와 그룹 ID 를 가진다. NIS 에 렬거된 모든 그룹들은 myproject 에 대한 항목뒤에 놓인다.

경고

+와 −의 특성들은 NIS 의 부분이다. 그러므로 NIS 가 설치되지 않았으면 이 특성들은 동작하지 않는다.

파일

/etc/group

/etc/logingroup

관련항목

groups(1), newgrp(1), passwd(1), setgroups(2), crypt(3c), getgrent(3c), initgroups(3c), passwd(4)

경고

/etc/passwd, /etc/group, /etc/logingroup 의 호환성을 보장하는데 리용될수 있는 도구는 없다. 그러나 pwck 와 grpck 가 그 과제를 단순하게 하는데 리용될수 있다(pwck(1M)과 grpck(1M)을 참고).

/etc/group 에 있는 그룹통과암호들을 설정하기 위한 도구는 없다.

표준일치

gorup:SVID2, SVID3, XPG2

inittab

inittab – 초기화데몬에 의하여 리용되는 파일

---

inittab(4)

이름

inittab – 부트(boot)초기화프로쎄스를 위한 스크립트

해설

/etc/inittab 파일은 일반적인 **프로쎄스발송자**(Process Dispatcher)로서의 역할을 수행하는 부트초기화데몬을 위한 스크립트를 제공한다(init(1M)을 참고). 부트초기화프로쎄스발송작용의 기본을 이루는 프로쎄스는 개별적인 **말단행**(Terminal Line)들을 초기화하는 행프로쎄스/usr/sbin/getty 이다. 특별히 부트초기화에 의하여 발송된 다른 프로쎄스들은 데몬들과 쉘들이다.

inittab 파일은 위치에 종속되는 항목들로 구성되며 다음과 같은 형식을 가진다.

id:rstate:action:process

매 항목은 행바꾸기로 끝난다. 그러나 행바꾸기앞에 \을 기입하면 그 항목의 계속임을 가리킨다. 매 항목은 1024 개까지의 기호들을 포함할수 있다. 프로쎄스마당에서 #로 시작되는것은 설명문이다(sh(1)을 참고).

gettys 를 파생하는 행들에 대한 설명문은 who 지령에 의해 현시된다(who(1)을 참고). 그것들은 행의 위치와 같은 정보들을 포함할수 있다. inittab 파일안의 항목들에는 제한이 있다.

id　항목을 유일하게 식별하는데 리용되는 1-4 개 기호값을 준다. 중복되는 항목들은 오유통보문을 발생하지만 그렇지 않을 때에는 무시된다. 항목들을 식별하는데 4 개 기호값을 리용할것을 권고한다.

rstate　이 항목이 처리되여야 할 실행준위를 정의한다. 실행준위들은 부트초기화에 의하여 파생된 매 프로쎄스에 하나 혹은 그이상의 실행준위들이 할당되는 체계에서 프로쎄스들의 조성에 대응한다. 실행준위들은 0-6 범위의 수에 의하여 표현된다. 실례로 체계가 실행준위 1 에 있으면 estate 마당에 1 을 가지는 항목들이 처리된다.

실행준위를 변화시키기 위하여 부트초기화가 요구될 때 rstate 마당에 목표실행준위에 대한 항목을 가지지 않는 모든 프로쎄스들이 경고신호(SIGTERM)를 보내며 제거신호(SIGKILL)에 의해 강압적으로 꺼지기전에 20 초주기로 허락한다. 한개이상의 실행준위값을 입력하여 한 프로쎄스에 대한 여러개 실행준위를 규정할수 있다. 실행준위가 규정되지 않으면 그 프로쎄스는 모든 실행준위 0-6 에 대하여 유효한것으로 가정된다.

다른 3 개의 값 a, b, c 들이 rstate 마당에 나타날수 있다. rstate 마당에서 이 기호들을 가지는 항목들은 사용자초기화프로쎄스가 그것들을 실행시키려고 할 때에만 처리된다(현재체계실행준위를 고려함이 없이). 그것들은 부트초기화가 전혀 들어 설수 없는 실행준위 a, b, c 와는 차이난다. 또한 이 프로쎄스들에 대한 실행요구는 현재수값실행준위를 변화시키지 않는다.

더우기 a, b, c 항목에 의하여 시동되는 프로쎄스는 부트초기화가 준위들을 변화시킬 때 제거되지 않는다. 프로쎄스는 inittab 에 있는 그 행이 action 마당에서 표식되지 않았을 때 또는 그 행이 inittab 로부터 삭제되였을 때 또는 부트초기화가 단일사용자상태에로 넘어 갔을 때 제거된다.

action　이 마당에 있는 실마리어는 process 마당에서 규정된 프로쎄스를 어떻게 처리하겠는가를 부트초기화에 알려 준다. 다음과 같은 작용들이 규정될수 있다.

boot　부트초기화를 위하여 inittab 파일의 항목을 처리한다. 부트초기화는 프로쎄스를 시동하고 그것이 꺼지는것을 기다리지 않는다. 그리고 그 프

로쎄스가 종결되였을 때 그것을 재시동하지 않는다. 이 지령이 의미를 가지려면 rstate 가 기정이든가 시동할 때 부트초기화의 실행준위와 들어 맞아야 한다. 이 작용은 체계의 하드웨어실행에 따르는 초기화기능에 대하여 쓸수 있다.

bootwait 부트초기화를 위하여 inittab 파일의 항목을 처리한다. 부트초기화는 프로쎄스를 시동하고 그것이 꺼 지기를 기다린다. 그리고 그 프로쎄스가 꺼 졌을 때 그 프로쎄스를 재시동하지 않는다.

initdefault 이 작용을 가진 항목은 부트초기화가 처음으로 진행될 때에만 검사된다. 부트초기화는 어느 실행준위를 초기에 리용할것인가를 결정하기 위하여 이 항목을 리용한다. rstate 마당에서 규정된 가장 높은 실행준위를 택하고 그것을 초기상태로 리용한다. rstate 마당이 비였으면 부트초기화는 실행준위 6 을 택한다. initdefault 항목은 부트초기화가 inittab 에서 initdefault 항목을 찾지 못하면 초기실행준위를 시동할 때 사용자에게 요구한다.

off 이 항목에 결합된 프로쎄스가 현재 실행중에 있으면 경고신호(SIGTERM)를 보내고 제거신호(SIGKILL)에 의해 프로쎄스가 강압적으로 꺼지기전에 20 초동안 기다린다. 그 프로쎄스가 존재하지 않으면 그 항목을 무시한다.

once 부트초기화가 rstate 항목에 들어 맞는 실행준위를 입력할 때 프로쎄스를 시동하고 그것이 꺼 지기를 기다리지 않는다. 그것이 제거되였을 때 그것을 재시동하지 않는다. 부트초기화가 새로운 실행준위를 입력하지만 그 프로쎄스는 여전히 실행준위에서 실행되고 있다면 그 프로쎄스는 재시동되지 않는다.

ondemand 이 지령은 respawn 작용과 의미가 같다. 이것은 실제적으로 respawn 과 기능적으로 같지만 실행준위와의 결합을 분리시키기 위하여 다른 실마리어를 붙였다. 이것은 rstate 마당에서 서술된 a, b, c 들과만 리용된다.

powerfail 이 항목과 결합된 프로쎄스는 부트초기화가 전원차단신호(SIGPWR)를 받았을 때에만 실행된다(signal(5)를 참고).

powerwait 이 항목과 결합된 프로쎄스는 부트초기화가 전원차단신호(SIGPWR)를 받았을 때에만 실행되며 inittab 의 임의의 프로쎄스를 계속하기전에 그것이 꺼 질 때까지 기다린다.

respawn 프로쎄스가 존재하지 않으면 프로쎄스를 시동하고 그것이 꺼 지기를 기다리지 않는다. (inittab 파일검사를 계속한다.) 그것이 꺼 지면 다시 시동한다. 프로쎄스가 현재 존재하면 아무것도 하지 않으며 inittab 파일검사를 계속한다.

sysinit 이 행의 항목들은 부트초기화가 콘솔접근을 시도하기전에 실행된다. 이 항목은 부트초기화가 실행준위정보를 얻으려고 시도하는 장치를 초기화하기 위해서만 리용된다.

wait 부트초기화가 그 항목의 rstate 에 들어 맞는 실행준위에 들어 갈 때 프로쎄스를 시동하고 그것이 꺼 지기를 기다린다. 부트초기화가 동일한 실행준위에 있을 때 inittab 파일을 계속 읽으면 이 항목은 무시된다.

process 이것은 실행되여야 할 sh 지령이다. 전체 process 마당은 exec 로 앞붙이되며 "sh ?c 'exec command' "로서 전달된다. 이 리유로 하여 exec 가 뒤따르는 sh 문장론은 process 마당에 나타날수 있다. 설명문은 ;#Comment 문장론을 리용하여 삽입될수 있다.

경고

4 개 기호로 구성된 id 를 리용할것을 권고한다. 많은 pty 봉사기들은 pty 이름의 마지막두개 기호를 id 로 리용한다. 만일 pty 봉사기에 의하여 선택된 id 가 inittab 파일에서 리용된것과 충돌된다면 /etc/utmp 파일은 낡은것으로 된다. 낡은 /etc/utmp 파일은 who 와 같은 지령들이 정확하지 못한 정보를 보고하게 할수 있다.

파일

/etc/inittab 부트초기화에 의하여 발송된 프로쎄스들의 파일

관련항목

sh(1), getty(1M), exec(2), open(2), signal(5)

mount

mount − 장비된 파일체계를 렬거하거나 또는 파일체계들을 장비한다.

---

mount(1M)

이름

mount(generic), umount(generic) − 파일체계들을 장비하거나 내리운다.

형식

/usr/sbin/mount [-1] [-p | -v]
/usr/sbin/mount -a [-F FStype] [-eQ]
/usr/sbin/mount [-F FStype] [-eQrV] [-o specific_options]
{special | directory}
/usr/sbin/mount [-F FStype] [-eQrV] [-o specific_options]
special directory
/usr/sbin/umount [-v] [-V] {special | directory}
/usr/sbin/umount -a [-F FStype] [-v]

해설

mount 지령은 파일체계들을 장비한다. 상급사용자만이 파일체계들을 장비할수 있다. 다른 사용자들은 mount 지령을 장비된 파일체계들을 렬거하는데 리용할수 있다.
mount 지령은 특수한 제거가능한 파일체계를 파일나무의 등록부에 붙여 준다. 이미 존재해야 하는 등록부는 새롭게 장비되는 파일체계뿌리의 이름으로 된다. special 과 directory 는 절대경로이름으로 주어 져야 한다. special 과 directory 가 생략되면 mount 는 생략된 그 값을 /etc/fstab 파일에 있는 항목으로 결정한다. mount 는 /를 제외한 임의의 제거가능한 파일체계상에 장비할수 있다.
mount 가 인수없이 실행되면 장비된 모든 파일체계들을 파일체계장비표 /etc/mnttab 로부터 렬거한다.
unmount 지령은 장비된 파일체계들을 내리운다. 상급사용자만이 파일체계들을 내리울수 있다.

선택항목들(mount)

mount 지령은 다음과 같은 선택항목들을 가진다.

-a /etc/fstab 에 수술된 모든 파일체계들을 장비하려고 시도한다. /etc/fstab 안의 모든 선택적인 마당들이 포함되고 지원되여야 한다. 선택항목 -F 가 규정되면 /etc/fstab 에 있는 모든 파일체계들이 장비된다. 파일체계들은 /etc/fstab 에 렬거된 순서로 장비되지 않아도 된다.

-e 장황한(verbose) 방식이다. 표준출력에 어느 파일이 장비되였는가를 가리키는 통보문을 기입한다.

-F FStype

조작을 수행할 파일체계형인 FStype 를 규정한다(FStype(1M)을 참고). 이 항목이 지령행에 포함되지 않으면 special 을 /etc/fstab 파일에 있는 항목과 정합을 하든가 또는 statfsdev( )

에 의하여 얻어 지는 special 의 파일체계통계로부터 파일체계형을 결정한다(statfsdev(3c)를 참고).

-l 작용들을 국부적인 파일체계에로 제한한다.

-o specific_options
매 파일체계형에 특정한 선택항목들을 규정한다. specific_options 는 그 지령의 FStype 특정의 방안을 의미하는 부분인수들 또는 실마리어와 속성의 쌍들이 반점으로 분리된 목록이다. 지원된 specific_options 의 해설에 대하여서는 FStype 특정의 지도서를 보면 된다.

-P 장비된 파일체계들의 목록을 /etc/fstab 형식으로 보고한다.

-Q 이미 장비된 파일체계를 다시 장비하려고 할 때 생겨 나는 오유통보문이 현시되는것을 방지한다.

-r 규정된 파일체계를 read_only 로 장비한다. 물리적으로 쓰기보호된 파일체계들은 이 방식으로 장비되여야 하며 그렇지 않으면 접근시간이 변경되였을 때 오유들이 발생된다.

-v 파일체계형과 기발들로 정규적인 출력을 보고한다. 그러나 directory 와 special 마당들은 전환된다.

-V 완성된 지령행을 출력한다. 그러나 다른 작용은 수행하지 않는다. 그 지령행은 사용자특정의 선택항목들과 /etc/fstab 로부터 유도된 다른 정보들을 포함하여 생성된다. 이 선택항목들은 사용자가 그 지령행을 확인해 볼수 있게 한다.

선택항목(umount)

umount 지령은 다음의 선택항목들을 가진다.

-a /etc/mnttab 에 서술된 모든 파일체계들을 내리운다. /etc/mnttab 에 있는 모든 선택적인 마당들이 포함되고 지원되여야 한다. FStype 가 규정되면 /etc/mnttab 에 있는 모든 파일체계들이 내리워 진다. 파일체계들이 /etc/mnttab 에 렬거된 순서로 내리워 질 필요는 없다.

-F FStype
조작을 수행할 파일체계형인 FStype 를 규정한다. 이 선택항목이 지령행에 포함되지 않으면 special 을 /etc/mntab 에 있는 항목과 정합하여 FStype 를 결정한다. 들어 맞는것이 없으면 그 지령은 실패한다.

-v 장황한 방식이다. 어느 파일체계가 내리워 져야 하는가를 가리키는 통보문을 표준출력에 기입한다.

-V 완성된 지령행을 출력한다. 그러나 다른 작용은 수행하지 않는다. 그 지령행은 사용자특정의 선택항목들과 /etc/fstab 로부터 유도된 다른 정보들을 포함하여 생성된다. 이 선택항목들은 사용자가 그 지령행을 확인해 볼수 있게 한다.

실례

현재 장비된 파일체계들을 렬거한다.

mount

HFS 파일체계 /dev/dsk/c1d2s0 을 등록부 /home 에 장비한다.

mount -F hfs /dev/dsk/c1d2s0 /home

동일한 파일체계를 내리운다.

umount /dev/dsk/c1d2s0

저자

mount 는 캘리포니아종합대학, HP, AT&T, 버클리, Sun Microsystems 에서 개발되였다.

파일

/etc/fstab 체계들에 대한 정적인 정보
/etc/mnttab 장비된 파일체계표

관련항목

mount_FStype(1M), mount(2), fstab(4), mnttab(4), fs_wrapper(5), quota(5)

표준일치

mount:SVID3
umount:SVID3

newgrp

newgrp - 그룹을 변경시킨다.

---

newgrp(1)

이름

newgrp - 새 그룹에로 절환한다.

형식

nergrp [-] [group]

해설

newgrp 지령은 사용자 ID 를 변경하지 않고 그룹 ID 를 변경하여 현재쉘을 새로운 쉘로 교체한다.

교체가 성과적으로 진행되려면 규정된 그룹이 존재하고 사용자 ID 가 그 그룹의 성원이든가 그룹이 통과암호를 가지며 그것을 말단으로부터 줄수 있어야 한다.

그룹을 생략하면 newgrp 는 통과암호파일 /etc/passwd 에서 규정된 그룹을 새로운 그룹으로 한다.

그룹이 성과적으로 변경되였든 아니든 또는 새 그룹이 낡은것과 동일하든 아니든 newgrp 는 현재쉘을 통과암호파일항목의 쉘마당에 규정된것으로 교체한다.

그 마당이 비였으면 newgrp는 POSIX쉘 /usr/bin/sh를 리용한다(sh-posix(1)을 참고).

-를 첫 인수로 규정하면 새로운 쉘은 사용자가 방금 가입한것처럼 시동된다. -를 생략하면 새로운 쉘은 부분쉘처럼 시동된다.

반출된 변수들은 자기의 값들을 유지하며 새로운 쉘에로 전달한다. 반출되지 않은 모든 변수들은 삭제된다. 그러나 새로운 쉘은 그것들을 기정값으로 재설정한다. 새로운 쉘이 시동될 때 현재프로쎄스가 교체되기때문에 새로운 쉘로부터 탈퇴하는것은 newgrp 이 실행된 파일로부터 탈퇴하는것과 동일한 효과를 가진다.

외부적영향

국제적인 코드모임보장

7 비트 USASCII 코드모임의 기호들이 그룹이름에서 지원된다(ascii(5)를 참고).

진단

newgrp 지령은 다음의 오유통보문을 준다.

Sorry	사용자 ID 는 그룹성원으로 될수 없다.
Unknown group	그룹이름은 /etc/group 에 존재하지 않는다.
Permission denied	통과암호가 요구되는 경우에 그것은 말단으로부터 온다.
You have no shell	표준입력은 말단파일이 아니므로 새로운 쉘은 실패한다.

실례

가입절차를 실행하지 않고 현재그룹으로부터 그룹사용자들에로 변경하려면 다음과 같이 한다.

```
newgrp users
```

현 그룹으로부터 그룹사용자들에로 변경하고 가입절차를 실행시키려면 다음과

같이 한다.

newgrp -users

경고

/etc/group 에 통과암호를 입력하기 위한 편리한 방법이 없다. 그룹통과암호는 실천적으로 안전성이 빈약하기때문에 리용하지 않는것이 좋다. 그룹통과암호들은 앞으로 개발될 HP-UX 에서는 소거될것이다.

파일들

/etc/group 체계그룹파일
/etc/passwd 체계통과암호파일

관련항목

csh(1), ksh(1), login(1), sh-bourne(1), sh-posix(1), group(4), passwd(4), environ(5)

표준일치

newgrp:SVID2, SVID3, XPG2, XPG3, XPG4

passwd

passwd − 사용자들에 대한 정보를 포함하는 파일이다. (이것은 사용자의 통과암호를 변경시키는데 리용되는 Passwd 지령과 차이난다.)

---

passwd(4)

이름

passwd − 통과암호파일, pwd.h

해설

passwd 는 매 사용자에 대하여 다음의 정보를 포함한다.

- 가입이름
- 코드화된 통과암호
- 수값사용자 ID
- 수값그룹 ID
- 식별에 리용될수 있는 예약된 마당

- 초기작업등록부
- 쉘로 리용할 프로그람

이것은 ASCII 파일이다. 매 사용자의 항목안에 있는 매 마당들은 두 점으로 분리된다. 사용자들은 줄바꾸기로 분리된다. 이 파일은 /etc 등록부에 들어 있다. 이 파일은 일반적인 읽기허락을 가질수 있으며 실례로 수값사용자 ID 를 이름으로 넘기는데 리용될수 있다. 통과암호마당이 null 일 때 체계가 신용 있는 체계로 변환되지 않았으면 통과암호는 요구되지 않는다.

쉘마당이 null 이면 /usr/bin/sh 이 리용된다.

코드화된 통과암호는 아래에 서술된 64 개 기호모임으로부터 선택된 13 개 기호들로 이루어 진다. 통과암호가 null 일 때에는 제외된다. 통과암호마당에 수자가 아닌 한 기호를 입력하면 가입은 차단된다.

수자들을 표현하는데 리용된 기호들은 0 대신에 . , 1 대신에 /, 2-11 대신에 0-9, 12-37 대신에 A-Z, 38-63 대신에 a-z 를 리용한다.

통과암호시효는 개별적사용자에 대하여 효력을 가진다. 통과암호파일에 있는 코드화된 통과암호는 뒤에 반점과 우의 자모들로 이루어 진 비지 않은 기호렬을 가진다. 이 기호렬은 통과암호시효를 실현하는데 필요되는 "나이"를 정의한다.

나이의 첫 기호 M 은 통과암호가 최대로 몇주동안 유효한가를 가리킨다. 통과암호의 기한이 지난 후에 가입하려고 하는 사용자는 새로운 통과암호를 정하여야 한다. 다음번 기호 m 은 통과암호가 변경되기전에 기한이 끝나야 할 최소주기를 가리킨다. 나머지 기호들은 통과암호가 마지막으로 변경된(빈 기호렬은 0 과 같다.) 주(1970 년부터 계산하여)를 정의한다. M 과 m 은 우에서 본 "수자"들의 64 개 기호의 모임에 대응하는 0-63 범위의 수값을 가진다. M=m=0(기호렬 . 또는 …으로부터 유도된)이면 사용자는 다음번에 가입할 때 통과암호를 변경시켜야 한다. M<m 이면(기호렬. /에 의해 표시된다.) 상급사용자만이(사용자는 아니고) 통과암호를 변경시킬수 있다. 특별히 신용 있는 체계들에서 사용자가 통과암호를 변경시키는것은 금지되여 있다.

신용 있는 체계들은 통과암호시효와 통과암호발생을 지원한다. 신용 있는 체계에로의 변환과 통과암호에 대하여 더 많은것을 알려면 HP-UX 체계관리과제안내서와 sam(1M)을 보면 된다.

getpwent(3c)는 마당들에 대한 값을 <pwd.h>에 소개된 다음과 같은 구조로 지적한다.

```
struct  passwd   {
  char       *pw_name;
  char       *pw_passwd;
```

```
    uid_t    pw_uid;
    gid_t    pw_gid;
    char     *pw_age;
    char     *pw_comment;
    char     *pw_gecos;
    char     *pw_dir;
    char     *pw_shell;
    aid_t    pw_audid;
      int    pw_audflg;
};
```

0~99 범위는 사용자와 그룹 ID 에 대하여 리용하지 말아야 한다. 그래야 체계쏘프트웨어에 할당할 ID 들이 충돌하지 않는다.

passwd 구조의 pw_gecos 마당에 보관된 사용자의 이름, 사무실의 위치, 생김새, 집의 전화는 chfn 지령(chfn(1)을 참고)을 리용하여 설정될수 있으며 finger(1)지령에 의하여 리용된다. 사용자의 실지 이름은 pw_gecos 마당에서 기호 &에 의하여 표현된다. 일부 봉사프로그람(finger 를 포함하여)들은 이 기호를 가입이름으로 치환하고 가입이름의 첫 문자를 대문자로 교체한다.

보호특성

신용 있는 체계들에서 매 사용자에 대한 코드화된 통과암호는 파일/tcb/files/auth/c/user_name(여기서 c 는 user_name 의 첫 문자이다.)에 보관된다. 통과암호정보파일들은 대역적으로 리용될수 없다. 코드화된 통과암호는 13 개이상의 기호들을 포함할수 있다. 실례로 사용자 david 에 대한 통과암호파일은 /tcb/files/auth/d/david 에 보관된다. 통과암호외에 /tcb/files/auth/c/user_name 에 있는 사용자정보는

- 수값결산 ID
- 수값결산기발

을 포함한다.

/etc/passwd 와 같이 이 파일은 ASCII 파일이다. 매 사용자항목안의 마당들은 두점으로 분리된다. 자세히 알려면 authcap(4)와 prpwd(4)를 보면 된다. /tcb/files/auth/c/*에 포함된 통과암호들은 /etc/passwd 의 코드화된 통과암호마당에 포함된것들보다 우선권을 가진다. 이 파일에 있는 코드화된 통과암호를 리용하여 사용자가 확증된다. passwd(1)의 SECURITY FEATURES 부분구획에서 서술된 통과암호시효기술은 이 통과암호에 리용된다.

망특성

NFS

passwd 파일은 첫 렬에서 + 또는 -로 시작하는 항목들을 가질수 있다. 그러한 행들은 망정보체계의 망자료기지에 접근하는데 리용된다. +로 시작되는 행은 망정보체계로부터의 항목들을 통합시키는데 리용된다. +항목에는 3 가지 형태가 있다.

+	망정보체계통과암호파일의 전체 내용을 그 위치에 삽입한다.
+name	name 에 대한 항목을 망정보체계로부터 그 위치에 삽입한다.
+@name	망그룹 name 의 모든 성원들에 대한 항목을 그 위치에 삽입한다.

+항목이 null 이 아닌 통과암호, 등록부, gecos 또는 쉘마당을 가지면 그것들은 망정보체계에 포함된것을 무시한다. 수값사용자 ID 와 그룹 ID 마당들은 무시될수 없다.

passwd 파일은 -로 시작되는 행들도 가질수 있다. 이것은 망정보체계로부터의 항목들을 금지시킨다. -항목에는 다음과 같은 2 가지 형태가 있다.

-name	name 에 대한 항목들을 허락하지 않는다.
-@name	망그룹 name 의 모든 성원들에 대한 항목들을 허락하지 않는다.

경고

사용자 ID(uid) 17 은 Pascal 언어조작체계를 위하여 예약되였다.
사용자 ID(uid) 18 은 BASIC 언어조작체계를 위하여 예약되였다.
이것들은 HP-UX 와 동일한 디스크상에서 공존할수 있고 300 파 400 계렬콤퓨터들에 대한 조작체계들이다. 이 uid 들을 다른 목적에 리용하면 파일전송의 공유가 금지되게 된다.

뿌리사용자(uid 0)에 대한 가입쉘은 /sbin/sh 여야 한다. sh, ksh, csh 와 같은 다른 쉘들은 모두 앞선 시동프로쎄스단계에서는 장비될수 없는 /usr 아래에 배치되여 있다. 뿌리사용자의 가입쉘을 /sbin/sh 가 아닌 값으로 변경시키면 체계는 기능을 수행할수 없다.

pw_gecos 마당에 있는 정보는 앞으로 이 마당을 리용할 때 충돌을 일으킬수 있다. pw_gecos 마당을 사용자식별정보를 보관하는데 리용하는것은 그 어느 공업표준에도 형식화되여 있지 않다. 앞으로의 표준은 이 마당을 다른 목적으로 정의할것이다.

다음의 마당들은 다음과 같은 기호제한을 가진다.

- 가입이름마당은 8 개 기호를 넘을수 없다.

- 초기작업등록부마당은 63개 기호를 넘을수 없다.
- 프로그람마당은 44개 기호를 넘을수 없다.
- 이 마당들이 우에서 규정된 제한보다 길면 결과는 예측할수 없다.

아래의 마당들은 다음과 같은 수값제한을 가진다.

- 사용자 ID는 ?2와 UID_MAX사이의 옹근수값이다.
- 그룹 ID는 0과 UID사이의 옹근수값이다.
- 이 값들중의 하나가 범위를 벗어 나면 함수 getpwent(3c)는 그 ID값을 UID_MAX에 재설정한다.

실례들

NFS 실례

/etc/passwd파일의 본보기는 다음과 같다.

```
root:3Km/o4Cyq84Xc:0:10:System  Administrator:/:/sbin/sh
joe:r4hRJr4GJ4CqE:100:50:Joe  User, Post  4A, 12345:/home/joe:/usr/bin/ksh
+john:
-bob:
+@documentation : no-login:
-@marketing:
+ : : :Guest
```

이 실례에는 망정보체계가 헝클어 진 경우에 생겨 나는 사용자 root와 joe에 대한 특정한 항목들이 있다.

- 망정보체계에서 사용자 john의 통과암호는 변경없이 리용된다.
- 사용자 bob에 대한 항목들은 무시된다.
- 망그룹 documentation에 있는 매 사람에 대한 통과암호마당은 불가능하게 된다.
- 망그룹 marketing에 있는 사용자들은 getpwent(3c)에 의하여 얻어 지지 않으며 따라서 체계에 가입할수 없다.
- 그외의 사람들은 보통의 통과암호, 쉘, 홈등록부를 가지고 가입할수 있다. 그러나 Guest의 pw_gecos를 가지고는 가입할수 없다.

NFS 경고

+와 -의 특성은 NFS의 기능이므로 NFS가 설치되지 않았으면 그것들은 동작하지 않는다. 또한 이 특성들은 /etc/passwd로만 동작한다. 체계가 신용 있는 체계로 변환되면 코드화된 통과암호들은 보호된 통과암호자료기지 /tcb/files/auth/*/*로

부터만은 접근할수 없다. 망정보체계자료기지에 있는 임의의 사용자항목도 보호된 통과암호자료기지의 항목을 가져야 한다.

uid 의 -2 는 NFS 에 의한 원격뿌리접근(remote root access)을 위하여 예약되였다. 보통의 uid 에 주어 진 pw_name 은 nobody 이다. Uid 들이 부호붙은 값으로 보관되여 있기때문에 아래의 정의는 사용자 nobody 를 정합하기 위하여 <pwd.h>에 포함되였다.

UID_NOBODY (-2)

파일

/tcb/files/auth/*/* 체계가 신용 있는 체계로 변환될 때 리용되는 보호된 통과암호자료기지

/etc/passwd HP-UX 에서 리용되는 표준통과암호

관련항목

chfn(1), finger(1), login(1), passwd(1), a64l(3C), crypt(3C), getprpwent(3), getpwent(3C), authcap(4), limits(5)

표준일치

passwd: SVID2, SVID3, XPG2

ps

ps − 프로쎄스상태를 보고한다.

---

ps(1)

이름

ps - 프로쎄스상태를 보고한다.

형식

ps [-adeflp] [-g grplist] [-p proclist] [-R prmgrplist]
[-t termlist] [-u uidlist]

XPG4 형식

ps [-aAcdefHjlP] [-C cmdlist] [-g grplist] [-G gidlist]
[-n namelist] [-o format] [-p proclist] [-R prmgrplist]
[-s sidlist] [-t termlist] [-u uidlist] [-U uidlist]

해설

ps는 선택된 프로쎄스들에 대한 정보를 인쇄한다. 어느 프로쎄스들을 선택하고 그것들에 대한 무슨 정보를 인쇄할것인가를 규정하기 위하여 아래의 항목들을 리용한다.

프로쎄스선택을 위한 인수들

프로쎄스를 선택하기 위하여 다음의 항목들을 리용한다.

주의: 어느한 항목이 기정환경(표준 HP-UX)과 XPG4 환경에서 리용되면 여기에 제공된 설명은 기정동작을 문서화한다. XPG4 동작에 대한 추가적인 정보를 더 알려면 EXTERNAL INFLUENCES 행의 UNIX95 변수를 보면 된다.

(none) 현재말단에 결합된 정보들을 선택한다.

-A (XPG4에서만)모든 프로쎄스들을 선택한다(-e에 대한 동의어).

-a 프로쎄스그룹자료기지들을 제외한 모든 프로쎄스들과 말단에 결합되지 않은 프로쎄스들을 선택한다.

-C cmdlist (XPG4에서만)cmdlist에 주어 진 토대이름을 가지고 지령을 수행하고 프로쎄스를 선택한다.

-d 프로쎄스그룹지휘자(process group leader)들을 제외한 모든 프로쎄스들을 선택한다.

-e 모든 프로쎄스들을 선택한다.

-g grplist 프로쎄스그룹지휘자가 grplist에 있는 프로쎄스들을 선택한다.

-G gidlist (XPG4에서만)실지그룹 ID 개수들과 그룹이름이 grplist에 주어 진 프로쎄스들을 선택한다.

-n namelist (XPG4에서만)이 항목은 무시된다. 이것의 존재는 표준승인에 대하여 허락된다.

-P proclist 프로쎄스 ID 번호들이 proclist에 주어 진 프로쎄스들을 선택한다.

-R prmgrplist

이름 또는 ID 번호들이 prmgrplist에 주어 진 PRM 프로쎄스자

료그룹에 속하는 프로쎄스들을 선택한다. DEPENDENCIES 를 참고.

-s sidlist　(XPG4 에서만)작업지휘자가 sidlist 에 주어 진 프로쎄스들을 선택한다(-g 에 대한 동의어).

-t termlist　termlist 에 주어 진 말단들에 결합된 프로쎄스들을 선택한다. 말단식별기호들은 다음의 두가지 형식 즉 장치파일이름(tty04 와 같은) 또는 장치파일이름이 tty 로 시작하는 경우에는 그것의 나머지(04 와 같은)로 규정될수 있다. 장치파일이 /dev 또는 /dev/pty 가 아닌 다른 등록부에 있다면 말단식별기호들은 장치파일을 포함하는 /dev 아래의 등록부이름을 포함해야 한다.

-u uidlist　효과적인 사용자 ID 번호 또는 가입이름들이 uidlist 에 주어 진 프로쎄스들을 선택한다.

-U uidlist　(XPG4 에서만)실지 사용자 ID 번호 또는 가입이름이 uidlist 에 주어 진 프로쎄스들을 선택한다.

선택항목 -a, -A, -d, -e 들중 하나가 규정되면 선택항목 -C, -g, -G, -p, -R, -t, -u, -U 들은 무시된다.

선택항목 -a, -A, -d, -e 들중 하나이상이 규정되면 최소한정선택항목들이 효력을 가진다. 선택항목 -C, -g, -G, -p, -R, -t, -u, -U 들중 하나이상이 규정되면 규정된 선택항목들에 들어 맞는 프로쎄스들이 선택된다.

선택항목 -C, -g, -G, -p, -R, -t, -u, -U 들이 인수로 리용되는 목록들은 다음과 같은 두가지 형식들중의 하나로 규정될수 있다.

- 식별기호들이 반점으로 분리된 목록
- 기호(")로 둘러 막히고 반점 또는 공백들로 분리된 식별들의 목록

출력형식선택항목들

어느 자료형들이 출력목록에 포함되는가를 조종하기 위하여 다음의 선택항목들을 리용한다.

(none)　기정 렬들은 순서대로 다음과 같다.

pid, tty, time, comm

-f	user, pid, ppid, cpu, stime, tty, time, args 렬들을 순서대로 보여 준다.
-l	flags, state, uid, pid, ppid, cpu, intpri, nice, addr, sz, wchan, tty, time, comm 렬들을 순서대로 보여 준다.
-fl	flags, state, user, pid, ppid, cpu, intpri, nice, addr, sz, wchan, stime, tty, time, args 렬들을 순서대로 보여 준다.
-c	(XPG4 에서만)CPU 와 렬 nice 를 제거한다. 렬 intpri 를 렬 cls 와 pri 로 교체한다.
-j	(XPG4 에서만)렬 pgid 와 sid 를 렬 ppid 의 뒤에 첨가한다.
-P	렬 prmid(-l 에 대한) 또는 prmgrp(-f 또는 ?fl 에 대한)을 렬 pid 의 앞에 첨가한다(DEPENDENCIES 를 참고).
-o format	(XPG4 에서만) format 는 현시할 행들을 순서대로 또는 공백으로 분리시킨 목록이다(유효한 렬이름들은 아래에 렬거되여 있다.). 렬이름뒤에는 = 또는 그 렬에 대한 머리부로 리용될 기호렬이 놓일수 있다(기호 =의 뒤에 있는 반점 또는 공백은 렬머리부의 부분으로 될수 있다. 더 많은 렬들이 요구되면 그것들은 추가적으로 선택항목 -o 로 규정되여야 한다.). 렬의 너비는 현시될 자료의 너비와 렬머리부의 너비보다 커야 한다. 렬머리부가 비면 머리부행은 인쇄되지 않는다. 이 선택항목은 선택항목 -c, -f, -j, -l, -P 들을 무시한다.
-H	(XPG4 에서만)프로쎄스계층을 보여 준다. 매 프로쎄스는 선조아래에 현시되는데 그 프로쎄스에 대한 렬 args 또는 comm 의 내용들은 선조들의것보다 들여 씌여 진다. 이 선택항목을 주면 기억자리와 속도에서 손해를 본다. 렬이름들과 그 의미들을 아래에서 보여 준다. 매 렬에 대한 기정머리부는 렬이름을 대문자로 표시한다.
addr	프로쎄스의 기억자리주소이다(상주하는 경우). 기억자리에 상주하지 않으면 디스크주소이다.
args	프로쎄스가 창조되였을 때 주어 지는 지령행이다. 이 렬은 요구되는 경우에 마지막으로 주어 져야 한다. 지령행의 부분모임만이 핵심부에 보관된다. 이 렬에 출력되는것은 공백을 포함할수 있다. 이 렬에 대한 기정머리부는 -o 가 규정된 경우에는 COMMAND 이고 그렇지 않은 경우에는 CMD 이다.

cls 프로쎄스의 일정을 세우는 클라스이다(rtsched(1)을 참고).

comm 지령이름이다. 이 렬에 출력된것은 공백을 포함할수 있다. 이 렬에 대한 기정머리부는 -o 가 규정된 경우에는 COMMAND 이고 그렇지 않은 경우에는 CMD 이다.

cpu 처리소자의 일정을 세우는 봉사프로그람이다. 이 렬에 대한 기정머리부는 C 이다.

etime 프로쎄스가 경과된 시간이다. 이 렬에 대한 기정머리부는 ELAPSED 이다.

flags 프로쎄스에 결합된 기발이다.

0	교환되였다.
1	핵심부에 있다.
2	체계프로쎄스이다.
4	핵심부에 잠겨 있다.
10	다른 프로쎄스에 의하여 추적되고 있다.
20	다른 추적기발이다.

이 렬에 대한 기정머리부는 f 이다.

intpri 핵심부에 의하여 내부적으로 보관된 프로쎄스의 우선권순위이다. 이 렬은 이미 존재하는것과의 호환성을 위하여 제공되였으며 그 리용은 장려되지 않는다.

gid 효과적인 프로쎄스소유자의 그룹 ID 번호

group 효과적인 프로쎄스소유자의 그룹이름

nice 우선권순위계산에서 리용되는 값이다(nice(1)을 참고). 이 렬에 대한 기정머리부는 NI 이다.

pcpu 마지막일정구간에서 프로쎄스에 의하여 리용되는 cpu 시간의 퍼센트이다. 이 렬에 대한 기정머리부는 %CPU 이다.

pgid 이 프로쎄스가 속하는 프로쎄스그룹의 ID 번호이다.

pid 프로쎄스의 ID 번호이다.

ppid 선조프로쎄스의 ID 번호이다.

pri 프로쎄스의 우선순위이다. 그 값의 의미는 프로쎄스의 일정을 세우는 클라스에 의존한다(우의 cls 와 rtsched(1)을 참고).

prmid PRM 프로쎄스자원그룹 ID 번호이다.

prmgrp　　PRM 프로쎄스자원그룹이름이다.

rgid　　실지프로쎄스소유자의 그룹 ID 번호이다.

rgroup　　실지프로쎄스소유자의 그룹이름이다.

ruid　　실지프로쎄스소유자의 사용자 ID 번호이다.

ruser　　실지프로쎄스소유자의 가입이름이다.

sid　　이 프로쎄스에 속하는 작업의 ID 번호이다.

state　　프로쎄스상태이다.

0	존재하지 않는다.
S	림시정지(sleeping)되여 있다.
W	대기(waiting)하고 있다.
R	실행(running)하고 있다.
I	중간(intermediate)에 있다.
Z	종결(terminated)되였다.
T	정지(stopped)되였다.
X	성장(growing)하고 있다.

이 렬에 대한 기정머리부는 S 이다.

stime　　프로쎄스의 시작시간이다. 경과된 시간이 24 보다 크면 그대신 시작날자가 현시된다.

sz　　프로쎄스의 핵심부영상의 물리적페지에 있는 크기이다. 본문, 자료, 탄창공간을 포함한다. 물리적페지의 크기는 머리부파일 <unistd.h>에서 _SC_PAGE_SIZE 에 의하여 정의되였다.

time　　프로쎄스에 대한 루적실행시간이다.

tty　　프로쎄스를 조종하는 말단이다. 이 렬에 대한 기정머리부는 -o 가 규정된 경우에는 TT 이고 그렇지 않은 경우에는 TTY 이다.

uid　　효과적인 프로쎄스소유자의 사용자 ID 번호이다.

user　　효과적인 프로쎄스소유자의 가입이름이다.

vsz　　프로쎄스의 핵심부영상의 크기를 바이트수로 준다(1024 바이트 단위)(우의 렬 sz 를 참고).

wchan　　프로쎄스가 대기하거나 림시정지되여 있는 경우이다. 아무것도 없으면 −가 현시된다.

주의

ps 는 프로쎄스가 창조되였을 때 주어 진 지령이름과 인수들을 인쇄한다. 만일 그 프로쎄스가 실행되는 도중에 자기 인수들을 변경시키면 그 변경은 ps 에 의하여 현시되지 않는다.

탈퇴하였으며 선조를 가지는 프로쎄스 그러나 아직 선조에 의하여 기다려 지는 프로쎄스는 <defunct>로 표식된다(exit(2)에서 zombie 프로쎄스를 참고).

렬 stime 에 인쇄되고 렬 etime 에 대한 계산에 리용되는 시간은 프로쎄스가 여러 갈래로 갈라 질 때의 시간이며 exec*()에 의하여 변경될 때의 시간이 아니다.

ps 출력을 안전하게 현시하고 쉽게 읽으려면 렬 comm 과 args 에 있는 모든 조종기호들을 현시하여야 한다.

외부적영향

환경변수들

UNIX95 는 이 지령에 대하여 XPG4 동작을 리용할것을 규정한다. 우에서 규정된 전체 선택항목모임을 그대로 두고 동작이 다음과 같이 되도록 XPG4 를 변경시킬 수 있다.

- TIME 렬형식은 mmmm:ss 로부터 [dd-]hh:mm:ss 로 변경된다.
- 마당 comm, args, user, prmgrp 들이 기정으로 포함되여 있거나 기발 -f 또는 -l 이 리용되면 그 마당들의 렬머리부들은 각각 CMD, CMD, USER, PRMGRP 로 변경된다.
- -a, -d, -g 는 프로쎄스그룹이 아니라 작업에 기초하여 프로쎄스들을 선택한다.
- -f 또는 -l 에 의하여 현시된 렬 uid 또는 user 은 실지사용자가 아니라 효과적인 사용자를 현시한다.
- 선택항목 -u 는 실지 UID 가 아니라 효과적인 UID 에 기초하여 사용자들을 선택한다.
- 선택항목 -C 와 -H 들은 XPG4 표준의 부분이 아니면 가능하게 된다.

LC_TIME 은 날자와 시간기호렬의 형식과 내용들을 설정한다. 만일 그것이 규정되지 않거나 null 이면 LANG 의 값을 기정으로 리용한다.

LANG 이 규정되지 않았거나 null 이면 C 를 규정으로 리용한다(lang(5)를 참고).

어느한 국제화변수가 무효한 설정값을 포함하면 모든 국제화변수들이 C 를 기정으로 한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트기호코드모임이 지원된다.

실례

현재 실행되고 있는 모든 프로쎄스들을 렬거하려면 다음과 같이 한다.

ps _ef

크론데몬과 같은 일정한 프로쎄스가 존재하는가를 보려면 지령이름 cron 에 대한 먼 오른쪽 렬을 검사한다.

ps -f -C cron

경고

ps 가 실행되고 있는 동안 어떤 변화가 일어 날수 있다. defunct 프로쎄스에 대하여 인쇄된 일부 자료는 적합치 않다.

말단들을 위한 두개의 특수한 파일들이 동일한 선택코드에 배치되여 있으며 그 말단들은 이름들로 보고된다. 사용자는 그 말단의 이름들을 리용하여 프로쎄스들을 선택할수 있다.

ps 의 사용자들은 정확한 마당너비와 그 출력의 간격을 믿지 말아야 한다. 왜냐하면 그것들은 체계들, HP-UX 의 방안, 현시될 자료에 따라 변하기때문이다.

종속성

HP 프로쎄스자원관리프로그람

선택항목 -P 와 -R 들은 설치되고 조성되여야 할 선택적인 HP 프로쎄스자원관리(PRM)쏘프트웨어를 요구한다. HP PRM 을 어떻게 서술하는가 하는것을 알려면 prmconfig(1)을, "프로쎄스자원그룹"에 대한 정의를 알려면 prmconf (4)를 보면 된다.

HP PRM 이 설치 및 조성되지 않고 -P 또는 -R 가 규정되였다면 경고통보문이 현시되며 렬 prmid 와 prmgrp 에는 -들이 현시된다.

파일

/dev 말단장치파일들의 등록부

/etc/passwd 사용자 ID 정보

/var/adm/ps_data 내부적인 자료구조

관련항목

kill(1), nice(1), acctcom(1M), exec(2), exit(2), fork(2), sysconf(2), unistd(5)

HP 프로쎄스자원관리프로그람 : HP 프로쎄스자원관리프로그람 사용자지도서에 있는 prmconfig(1), prmconf(4)

표준일치

ps: SVID2, XPG2, XPG3, XPG4

shutdown

shutdown – 실행되고 있는 모든 프로쎄스들을 정돈된 순서로 끈다.

---

shutdown(1M)

이름

shutdown - 모든 프로쎄스들을 끈다.

형식

/sbin/shutdown [-h| -r] [-y] [-o] [grace]

해설

shutdown 지령은 HP-UX 체계조작절차의 부분이다. 이것의 선차적인 기능은 현재 실행되고 있는 모든 프로쎄스들을 순서대로 조심스럽게 끄는것이다. shutdown 은 여벌복사 또는 체계일관성검사를 위하여 체계를 단일사용자방식에 놓으며 체계를 정지시키거나 재시동하는데 리용될수 있다. shutdown 은 대화형프로그람이다.

선택항목들과 인수들

shutdown 은 다음과 같은 선택항목들과 인수들을 가진다.

-h 체계를 끄고 정지한다.

-r 체계를 끄고 자동적으로 재시동한다.

-y 사용자로부터의 응답을 요구하지 않는다(사용자가 shutdown 프로쎄스와 대화하겠는가 하는것과 같은 모든 질문들에 yes 또는 no 로 응답한다).

-o　　디스크가 없는 클라스터환경에 있는 클라스터봉사기상에서 실행될 때 봉사기만을 끄며 의뢰기들을 재시동하지 않는다. 이 인수가 주어 지지 않으면 봉사기가 꺼질 때 모든 의뢰기들은 기정동작으로 재시동된다.

grace　　체계를 끄기전에 사용자들이 가입을 끝낼 주기(grace period)의 지속시간을 초수로 규정하는 10 진옹근수이거나 단어이다. 기정은 60 이다. grace 가 0 이거나 now 이면 shutdown 은 더 고속으로 실행된다.

-r(재시동)와 -h(정지)가 규정되지 않으면 **단독체계**(Standalone)와 봉사기체계들은 단일사용자상태에 놓인다. 의뢰기에 대하여 -r 또는 -h 가 규정되여야 한다면 단일사용자상태에로의 shutdown 은 허용되지 않는다(dcnodes(1M)과 init(1M)을 참고).

닫기절차

닫기는 다음의 단계들을 거친다.

- 환경변수 PATH 를 /usr/bin:/usr/sbin:/sbin 에 재설정한다.
- IFS 환경변수를 공백, tab, newline 으로 재설정한다.
- 사용자에게 shutdown 지령을 실행시킬 권한이 있는가를 검사한다. 권한이 있는 사용자들만이 shutdown 지령을 실행시킬수 있다. 허락권한파일 /etc/shutdown 에 대한 정보를 더 알려면 FILES 를 보면 된다.
- 현재작업등록부를 뿌리등록부(/)로 변경한다.
- 모든 파일체계들의 블로크들이 갱신된다(sync(1M)을 참고). 이것은 파일체계의 존재성을 확신하기 위하여 체계를 재시동하기전에 진행되여야 한다.
- 실지사용자 ID 는 상급사용자의 것으로 설정된다.
- 체계상에 현재 가입되여 있는 모든 사용자들에게 작업을 끝낼것을 알리는 통보문이 전송된다. 관리자는 이때 통보문을 규정할수 있다. 그렇지 않으면 표준경고통보문이 현시된다.
- 다음번 걸음은 체계가 단독기인가, 봉사기인가, 의뢰기인가에 의존한다.
- 체계가 단독이면 부분체계로 보고 파일체계를 내리우며 체계를 실행준위 0 에 가져 오기 위한 과제들을 수행하기 위하여 /sbin/rc 가 실행된다.
- 체계가 봉사기이면 클라스터에 있는 모든 의뢰기들은 재시동되여야 하는가를 결정하기 위하여 -o 인수가 리용된다. 기정동작은 /sbin/reboot 를 리용하여 모든 의뢰기들을 재시동하는것이다. -o 를 리용하면 봉사기만이 재시동되며 의뢰기들은 홀로 남아 있다. 그 다음 부분체계를 끄고 파일체계를 내리우며 체계를 실행준위 0 에 가져 오기 위한 과제들을 수행하기 위하여 /sbin/rc 가 실행된다.
- 체계가 의뢰기이면 체계를 실행준위 2 에 가져 오기 위하여 /sbin/rc 가

실행되며 그다음 /sbin/reboot 가 실행된다. 의뢰기는 단일사용자상태에 놓일수 없다.

- 선택항목 -h 또는 -r 가 선택되면 /sbin/reboot 의 실행에 의하여 체계가 재시동되거나 정지한다. 체계가 클라스터의뢰기가 아니고 단일사용자상태에 놓이지 않았다면 상태를 변경시키도록 초기화프로쎄스에 신호가 전송된다(init(1M)을 참고).

진단

device busy

이것은 가장 흔히 맞다드는 오유진단인데 개별적파일체계가 내리워 질수 없을 때 일어 난다(mount(1M)을 참고).

user not allowed to shutdown this system

사용자는 체계를 끌 권한이 없다. 사용자와 체계는 권한파일 /etc /shutdown.allow 에 있어야 한다.

실례

체계를 즉시 재시동하고 HP-UX 를 다시 실행시킨다.

```
shutdown  -r  0
```

대화적인 질문과 대답이 없이 체계를 5min(300s)내에 정지시킨다.

```
shutdown -h -y  300
```

10min 내에 실행준위 s 로 이행한다.

```
shutdown 600
```

파일

/etc/shutdwon.allow

권한파일

이 파일은 체계호스트이름과 체계를 재시동하거나 정지시킬 권한이 있는 사용자의 가입이름으로 이루어 지는 행을 포함한다.

상급사용자의 가입이름은 shutdown 을 실행시키기 위하여 이 파일에 포함되여야 한다. 그러나 파일이름이 없거나 길이가 0 이면 뿌리사용자는 shutdown 프로그람을 실행시킬수 없다.

이 파일은 유지를 목적으로 체계를 단일사용자상태에로 넘기는 권한에 영향을 주지 않는다. 이 조작은 상급사용자가 수행하려고 할 때에만 허락된다.

행의 앞에 있는 설명문기호 #는 그 행의 나머지를 무시하게 한다. 빈행도 무시된다. (설명문은 추가적인 기호를 붙여야 여러 행에서 계속될수

있다.)

통용기호 +는 모든 호스트와 모든 사용자들을 규정하기 위하여 호스트이름 또는 사용자이름의 자리에서 리용될수 있다(hosts.equiv(4)를 참고).

실례

```
# user1 can shut down systemA and systemB
systemA user1
systemB user1
# root can shut down any system
+ root
# Any user can shut down systemC
systemC +
```

경고

파일 shutdown.allow 에 있는 항목과 비교되는 사용자이름은 getlogin( )를 리용하여 얻어 지며 이것이 실패할 때에는 getpwuid( )를 리용하여 얻어 진다(getlogin(3)과 getpwuid(3)을 참고).

/etc/shutdown.allow 에 있는 호스트이름은 gethostbyname()를 리용하여 얻어진 호스트이름과 비교된다(gethostbyname(3)을 참고).

shutdown 은 등록부 /와 같은 뿌리기록권상의 등록부로부터 실행되여야 한다. 전송될수 있는 최대 통보문은 대략 970 개 기호이다.

NFS 디스크가 없는 클라스터봉사기상에서 shutdown 을 실행시킬 때 선택항목 -o 를 주지 않으면 봉사기의 의뢰기들은 재시동된다. 클라스터가 꺼질 때 개별적으로 재시동되거나 꺼지는 의뢰기는 없다.

관련항목

dcnodes(1M), fack(1M), init(1M), killall(1M), mount(1M), rebo-ot(1M), sync(1M), dcnodes(3), gethostbyname(3), getpwuid(3), ho-sts.equiv(4)

vipw

vipw – 자물쇠가 있는 passwd 파일을 편집한다.

---

vipw(1M)

이름

vipw – 통과암호파일을 편집한다.

형식

vipw

해설

vipw 는 알맞는 자물쇠를 설정하여 통과암호파일을 편집하며 통과암호파일의 자물쇠를 연 후에 필요한 처리를 수행한다. 이미 편집된 통과암호파일을 후에 다시 수정할수 있다. 환경변수 EDITOR 가 다른 편집기를 가리키지 않을 때에는 vi 편집기가 리용된다. vipw 는 뿌리에 대한 통과암호항목의 일관성검사를 수행하며 틀리게 형식화된 뿌리항목을 가진 통과암호파일이 설치되는것을 허락하지 않는다.

경고

체계가 재시동된후에 vipw 를 리용하여 파일 /etc/passwd 의 편집을 방해하는 체계파괴가 있을 때 파일 /etc/passwd.tmp 는 제거되지 않는다.

저자

vipw 는 캘리포니아종합대학과 버클리에서 개발되였다.

파일

/etc/passwd.tmp

관련항목

passwd(1), passwd(4)

제１４장. UNIX 수행도구들에 대한 개괄

체계에서는 수행해석을 위한 몇가지 방법들가운데서 한가지 방법을 택할수 있다. 이 방법의 선택은 마음대로 빨리 할수 있으나 그 방법이 산생시킨 자료의 해석을 시작하기 전에 주별로 혹은 월별로 수행시킬것을 요구하는 프로그람들의 오랜 기간의 작업계획화를 하는데는 일정한 시간이 걸린다. 이 장은 HP-UX 11i 체계를 포함하여 여러 체계들에서 수행시킨 실례들을 주고 있다. 일부 독자들이 HP-UX 와는 다른 UNIX 방안에 습관되여 있을수 있기때문에 HP-UX 체계와 다른 체계의 실례들도 포함시켰다. 이 장에서 HP-UX 의 대부분 실례들은 11i 체계를 반영하여 갱신되였다. 그러나 11.0 과 11i 에서 이 수행실례들사이의 차이점에 대하여서는 알지 못할수도 있다.

이 장에서는 공통으로 리용될 UNIX 의 일부 지령들과 여러가지 UNIX 방안에서 수행할 개선된 **도구**(Tool)들에 대하여 고찰한다. 여기서 론의된것들은 UNIX 지령들의 완전한 목록과 수행관리에 관계된 도구들에는 관심을 두지 않을것이다. 그렇지만 개괄을 주는데는 충분히 좋은 정보를 제공한다. 자기가 리용하는 UNIX 체계는 추가적지령들을 제공할수도 있고 개선된 수행의 해석도구를 가질수도 있다. 이 장에서는 대부분 공통적으로 리용되는 UNIX 의 수행과 관련된 지령들의 실례를 포함하여 수행해석의 일반적인 개괄을 준다.

표준 UNIX 지령

먼저 체계에 대한 어떤 통보를 받으려고 할 때 UNIX 입력재촉문에서 줄수 있는 일부 지령들을 보기로 한다. 이런 지령들은 다음과 같은 지령들을 포함한다.

- iostat
- vmstat
- netstat
- ps
- kill
- showmount
- swapinfo 와 swap
- sar

먼저 이 매개 지령들을 리용할 때 지령으로 산생한 출력정보의 리해를 도와 주며 이 출력을 어떻게 리용할것인가를 보기로 하자. 이 장의 마감에서는 많은 지령들을 편람으로 제시하고 있다.

UNIX 의 방안에 따라 이 지령들의 출력정보는 차이날수도 있다. 대부분의 UNIX 방안에서 산생된 기본정보는 같지만 출력형식은 좀 다를수 있다. 이리하여 보통 출력정보

만을 보려고 한다면 그 출력형식은 부차적인것으로 될것이다. 그러나 만일 이 출력정보를 어떤 방법으로 접수하여 그 정보를 처리하는 프로그람에서 리용하려고 한다면 출력형식은 중요한것으로 될것이다.

iostat에 의한 I/O과 CPU의 통계작성

iostat 지령은 CPU(중앙처리장치)가 I/O에 출력하는 작업의 성공정도를 지적하는 정보를 표시해 주며 **디스크**(disk)와 말단장치에 진행한 I/O 연산의 총 수를 표시해 준다. iostat는 유용한 모든 정보를 제공한다. 그러나 이 지령은 UNIX의 방안에 따라 약간씩 다르게 작용한다. 다음 실례들은 solaris체계, HP-UX체계, AIX체계에서 iostat가 출력한 결과를 보여 준다. iostat는 Linux 체계에서 제공되지 않고 있지만 이 장에서는 리용하기로 한다. 어떤 체계들에서나 말단통보를 보려면 선택항목 -t를 리용하여야 하는데 일부체계에서는 완전한 출력통보를 준다. 그러므로 자기의 UNIX 방안에 맞는 필요한 가장 좋은 선택항목들을 결정해 주어야 할것이다. 다음의 실례들은 지령 iostat의 결과를 보여주고 있다.

아래에서는 5s 간격으로 10번 수행된 solaris의 실례를 보여 준다.

\# **iostat 5 10**

tty		fd0			sd1			sd3			sd6			cpu			
tin	tout	kps	tps	serv	kps	tps	serv	kps	tps	serv	kps	tps	serv	us	sy	wt	id
0	0	0	0	0	0	0	0	3	0	57	0	79	0	0	7	49	43
0	47	0	0	0	0	0	0	14	2	75	0	0	0	0	2	0	98
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	98
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	98
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	99
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
0	16	0	0	0	0	0	0	6	1	35	0	0	0	0	4	0	96
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100

HP-UX실례는 5s 간격으로 5번 수행된 선택항목 -t를 포함한 경우에 해당된다.

\# iostat -t 5 5

tty		cpu			
tin	tout	us	ni	sy	id
1	58	5	1	10	84

```
device        bps      sps      msps
c1t2d0          0      0.0       1.0

               tty                     cpu
          tin    tout            us    ni    sy    id
            0      30             0     2    26    72

device        bps      sps      msps
c1t2d0        484    249.6       1.0

               tty                     cpu
          tin    tout            us    ni    sy    id
            0      31             1     3    23    73

device        bps      sps      msps
c1t2d0        517    256.1       1.0

               tty                     cpu
          tin    tout            us    ni    sy    id
            0      35             0     2    23    75

device        bps      sps      msps
c1t2d0        456    254.4       1.0

               tty                     cpu
          tin    tout            us    ni    sy    id
            0     744             1     6    38    55

device       bps      sps      msps
c1t2d0       155      83.1     1.0
#
```

아래에서는 5s 간격으로 10번 수행된 AIX 실례를 보여 준다.

```
# iostat  5  10
tty:        tin          tout     avg-cpu:   % user   % sys   %idle     %iowait
            0.0          0.0                    0.3      1.0    98.4         0.3
Disks:        % tm_act       Kbps      tps      Kb_read        Kb_wrtn
hdisk0          0.4            2.7      0.4      2366635          959304
```

```
hdisk1      0.0         0.0     0.0     18843       37928
hdisk2      0.1         0.6     0.1     269803      423284
hdisk3      0.0         0.0     0.0     20875       172
cd0         0.0         0.0     0.0     14          0

tty:    tin     tout    avg-cpu:  % user  % sys  %idle  %iowait
        0.0     108.2              0.0     0.2    99.8    0.0
Disks:      % tm_act    Kbps    tps     Kb_read     Kb_wrtn
hdisk0      0.0         0.0     0.0     0           0
hdisk1      0.0         0.0     0.0     0           0
hdisk2      0.0         0.0     0.0     0           0
hdisk3      0.0         0.0     0.0     0           0
cd0         0.0         0.0     0.0     0           0

tty:    tin     tout    avg-cpu:  % user  % sys  %idle  %iowait
        0.0     108.4              0.2     0.8    99.0    0.0
Disks:      % tm_act    Kbps    tps     Kb_read     Kb_wrtn
hdisk0      0.0         0.0     0.0     0           0
hdisk1      0.0         0.0     0.0     0           0
hdisk2      0.0         0.0     0.0     0           0
hdisk3      0.0         0.0     0.0     0           0
cd0         0.0         0.0     0.0     0           0
tty:    tin     tout    avg-cpu:  % user  % sys  %idle  %iowait
        0.0     108.4              0.4     0.2    99.4    0.0
Disks:      % tm_act    Kbps    tps     Kb_read     Kb_wrtn
hdisk0      0.0         0.0     0.0     0           0
hdisk1      0.0         0.0     0.0     0           0
hdisk2      0.0         0.0     0.0     0           0
hdisk3      0.0         0.0     0.0     0           0
cd0         0.0         0.0     0.0     0           0
tty:    tin     tout    avg-cpu:  % user  % sys  %idle  %iowait
        0.0     108.2              0.4     0.6    99.0    0.0
Disks:      % tm_act    Kbps    tps     Kb_read     Kb_wrtn
hdisk0      0.0         0.0     0.0     0           0
hdisk1      0.0         0.0     0.0     0           0
hdisk2      0.0         0.0     0.0     0           0
hdisk3      0.0         0.0     0.0     0           0
cd0         0.0         0.0     0.0     0           0
```

```
tty:      tin       tout     avg-cpu:   %  user   %  sys    %idle     %iowait
          0.0       108.4                  0.0       0.4      99.6        0.0
Disks:        %  tm_act      Kbps      tps      Kb_read       Kb_wrtn
hdisk0        0.0            0.0       0.0        0             0
hdisk1        0.0            0.0       0.0        0             0
hdisk2        0.0            0.0       0.0        0             0
hdisk3        0.0            0.0       0.0        0             0
cd0           0.0            0.0       0.0        0             0

tty:      tin       tout     avg-cpu:    % user    % sys    %idle     %iowait
          0.0       108.4                  0.6       0.0      99.4        0.0
Disks:        %  tm_act      Kbps      tps      Kb_read       Kb_wrtn
hdisk0        0.0            0.0       0.0        0             0
hdisk1        0.0            0.0       0.0        0             0
hdisk2        0.0            0.0       0.0        0             0
hdisk3        0.0            0.0       0.0        0             0
cd0           0.0            0.0       0.0        0             0

tty:      tin       tout     avg-cpu:    % user    % sys    %idle     %iowait
          0.0       108.2                  0.2       0.8      99.0        0.0
Disks:        %  tm_act      Kbps      tps      Kb_read       Kb_wrtn
hdisk0        0.0            0.0       0.0        0             0
hdisk1        0.0            0.0       0.0        0             0
hdisk2        0.0            0.0       0.0        0             0
hdisk3        0.0            0.0       0.0        0             0
cd0           0.0            0.0       0.0        0             0

tty:      tin       tout     avg-cpu:    % user    % sys    %idle     %iowait
          0.0       108.4                  0.4       0.0      99.6        0.0
Disks:        %  tm_act      Kbps      tps      Kb_read       Kb_wrtn
hdisk0        0.0            0.0       0.0        0             0
hdisk1        0.0            0.0       0.0        0             0
hdisk2        0.0            0.0       0.0        0             0
hdisk3        0.0            0.0       0.0        0             0
cd0           0.0            0.0       0.0        0     0

tty:      tin       tout     avg-cpu:    % user    % sys    %idle     %iowait
          0.0       108.4                  0.4       0.4      99.2        0.0
Disks:        %  tm_act      Kbps      tps      Kb_read       Kb_wrtn
```

hdisk0	0.0	0.0	0.0	0	0
hdisk1	0.0	0.0	0.0	0	0
hdisk2	0.0	0.0	0.0	0	0
hdisk3	0.0	0.0	0.0	0	0
cd0	0.0	0.0	0.0	0	0

여기서는 말단장치, CPU와 설치된 파일체계에 관하여 iostat로 받을 통보에 대한 설명을 주기로 한다. 통보는 약간 다르기때문에 HP-UX 출력에 대한 상세한 정보만을 포함시킬것이다. 이 마당들의 더 자세한 설명은 이 장의 마감에 있는 iostat의 편람에 제시하였다. 대부분의 마당들은 출력에 나타난다. 그런데 지령의 출력은 UNIX의 방안에 따라 약간 다르다.

접속된 매 말단장치(tty)에 대하여 "tin"과 "tout"를 출력한다. 이것들은 각각 두개의 말단장치로부터 읽은 기호의 개수를 표시하며 자기의 말단장치에 쓴 기호의 개수도 표시한다.

CPU 에 대하여 사용자방식(us)에서 소비한 시간의 퍼센트, 낮은 우선권(좋다고 하는 (ni) 낮은 우선권)에서 사용자과제를 수행하는데 소비된 시간의 퍼센트, 체계방식(sy)에서 소비된 시간의 퍼센트, CPU 가 놀고 있는 (id)시간의 퍼센트를 볼수 있을것이다.

또한 국부적으로 설치된 매 파일체계들에 대하여 초당 KByte (bps)로 전송된 정보, 초당 찾기회수(sps)와 평균찾기당 미리초(msps)를 받을것이다. NFS 가 설치된 디스크 혹은 자기의 봉사기의 **의뢰기마디점**(Client Node)에 있는 디스크에 대한 통보는 받지 못할것이다. iostat는 국부적으로 설치된 파일체계들에 대해서만 통보한다.

iostat의 출력을 볼 때 더 언급해야 할 파라메터들도 있다.

첫째로, 자기의 CPU 는 4 개의 부류가운데서 한 부류에서 보내고 있는 시간을 표시한다. CPU 의 통보는 선택항목 -t 로 표시한다. CPU 가 실지로 대부분의 시간을 놀고 있었다는것을 지적하는 id 의 수값이 크기때문에 **체계관리자**(Administrator)는 CPU 작업효과가 적다고 판정하는데 이것은 **낮은 성능**(Poor Performance)으로 체계는 작업하고 있다는것을 의미한다. 만일 CPU 가 대부분 놀고 있다면 그 원인은 **병목**(Bottleneck)이 CPU 가 아니라 I/O 혹은 기억 혹은 망으로 될수 있다는데 있다. 만일 CPU 가 대부분의 시간을 긴장하게 작업하고 있었다면(id 가 대단히 작다면) 모든 프로쎄스들은 언제나 좋게 수행하고 있다는것을 알수 있다(ni 수값의 검사). 이것은 CPU 의 많은 시간을 소비하는 몇개의 배경프로쎄스들이 있다면 좋은 수행으로 넘어 갈수 있다는것을 말하여 준다.

둘째로, 전송한 회수를 분석해야 한다. 보통 초당 블로크수(bps), 초당전송회수(tps), 초당 찾기회수(sps)와 같은것으로 지적될것이다. 이 수값들은 디스크에서 작업한 총 수를 지적한다. 만일 어떤 한 장치에 대한 값이 다른 장치에 대한 값보다 훨씬 더 크다면 총체적으로 작업부하의 량은 무질서하게 수행될수 있다. HP-UX 에서는 모든 디스크에서 평균 찾기당 미리초(msps)가 항상 1과 같다는것을 지적한다.

vmstat에 의한 가상기억의 통계작성

vmstat 는 **가상기억**(Virtual Memory)의 통계자료를 제공한다. 이 통계자료에서 **프로쎄스상태**(Status Of Process), 가상기억, 페지화성공 및 실패와 CPU 시간의 퍼센트는 프로쎄스상태에 대한 통보를 제공한다. vmstat 는 UNIX 의 방안에 따라 약간 다르게 작업한다. 다음의 실례들은 solaris 체계, HP-UX 체계, AIX 체계, Linux 체계에서 vmstat 의 출력정보를 보여 준다. 물론 자기의 UNIX 방안에서는 필요한 좋은 선택항목들을 결정해 주어야 한다. 다음의 실례들에서 출력정보는 5s 간격으로 10 번 진행된것을 제시한것이다. vmstat 지령에서 첫째 인수는 시간구간이고 둘째 인수는 출력이 진행될 회수이다.

Solaris 실례:

vmstat 5 9

procs			memory		page							disk				faults			cpu		
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	f0	s1	s3	s6	in	sy	cs	us	sy	id
0	0	0	4480	4696	0	0	1	0	0	0	0	0	0	0	79	864	130	297	0	7	92
0	0	0	133020	5916	0	3	0	0	0	0	0	0	0	3	0	102	42	24	0	2	98
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	70	48	24	0	0	100
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	74	42	24	0	0	100
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	35	45	23	0	0	99
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	65	66	26	0	0	100
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	52	44	23	0	1	99
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	0	0	53	54	24	0	1	99
0	0	0	133020	5916	0	0	0	0	0	0	0	0	0	1	0	60	53	25	0	2	98

HP-UX 실례:

vmstat 5 9

procs			memory		page							faults			cpu		
r	b	w	avm	free	re	at	pi	po	fr	de	sr	in	sy	cs	us	sy	id
5	240	0	17646	3979	2	0	0	0	0	0	0	0	778	193	17	3	80
4	242	0	16722	4106	0	0	0	0	0	0	0	814	20649	258	89	10	2
4	240	0	16649	4106	0	0	0	0	0	0	0	83	18384	218	91	9	0
4	240	0	16468	4106	0	0	0	0	0	0	0	792	19552	273	89	11	1
5	239	0	15630	4012	9	0	0	0	0	0	0	804	18295	270	93	8	-1
5	241	0	16087	3934	6	0	0	0	0	0	0	920	21044	392	89	10	0
5	241	0	15313	3952	11	0	0	0	0	0	0	968	20239	431	90	10	0
4	242	0	16577	4043	3	0	0	0	0	0	0	926	19230	409	89	10	0
6	238	0	17453	4122	0	0	0	0	0	0	0	837	19269	299	89	9	2

AIX 실례:

```
martyp $ vmstat 5 9
kthr     memory     page       faults        cpu
r  b   avm   fre  re  pi  po   fr   sr   cy    in   sy  cs us sy id wa
0  0  16604  246  0   0   0    0    2    0    149   79  36  0  1 98  0
0  0  16604  246  0   0   0    0    0    0    153  125  41  0  0 99  0
0  0  16604  246  0   0   0    0    0    0    143   83  33  0  0 99  0
0  0  16604  246  0   0   0    0    0    0    140   94  35  0  1 99  0
0  0  16604  246  0   0   0    0    0    0    166   62  32  0  0 99  0
0  0  16604  246  0   0   0    0    0    0    150  102  38  1  0 99  0
0  0  16604  246  0   0   0    0    0    0    183   78  34  0  0 99  0
0  0  16604  246  0   0   0    0    0    0    132   87  33  0  1 99  0
0  0  16604  246  0   0   0    0    0    0    147   84  38  0  0 99  0
```

Linux 실례:

```
# vmstat 5 5
  procs                       memory    swap      io     system       cpu
r  b  w  swpd  free  buff  cache  si  so  bi  bo   in    cs   us  sy  id
1  0  0  9432  1160  656   12024   1   2  14   1  138   274    3   1  96
1  0  0  9684   828  652   12148   0  50   0  14  205  8499   82  18   0
1  0  0  9684   784  652   11508   0   0   0   1  103  8682   81  19   0
1  0  0  9684   800  652   10996   0   0   0   0  101  8683   80  20   0
0  0  0  9772   796  652    9824  12  18   3   4  160  6577   66  17  18
```

지령 vmstat 로서 일정하게 많은 가치 있는 정보를 얻을수 있다. 여기서는 vmstat 로 얻어 진 정보를 부류별로 간단히 해설한다. 이 장의 마감에 HP-UX 의 편람이 있으므로 일부 마당들의 해설들을 참고하기 바란다. 앞의 실례에서 그 출력자료는 대단히 류사하다는것을 알수 있을것이다.

프로쎄스들은 다음과 같은 세 부류들중의 한 부류에 속할것이다. 즉 수행할수 있는 프로쎄스(r), I/O 로 봉쇄되거나 짧은 기간동안 자원에 의하여 봉쇄된 프로쎄스(b), 교환되는 프로쎄스(w)이다. 다음으로 기억에 대한 정보를 볼수 있다. avm 은 마지막 20 초동안에 수행되는 프로쎄스들이 소속된 가상기억페지의 개수이다. 만일 이 수가 론리적기억크기에서 핵심부크기를 던 값과 대략 같다면 거의 **강제적페지화**(Forced Paging)라는것을 의미한다. 렬 Free 는 체계의 자유페지목록에서 페지의 개수를 지적한다. 이 프로쎄스는 수행을 끝내고 다시 접근할수 없다는것을 의미하지는 않는다. 이것은 이 페지들을 최근에 접근하지 않았다는것을 의미한다. 이 렬은 무시해도 된다.

다음으로 **능동적페지화**(Active Paging)이다. 첫 마당(re)은 교환된 페지들을 보여 준

다. 이 페지들은 자유페지목록에서 취하지만 후에 리용되고 반환된다.

다음과 같은 세개의 부류는 상태의 개수를 보여 준다. 즉 세개의 부류는 보통 기구장치로부터 오는 초당 중단개수(in), 초당 체계호출개수(sy), 초당 문맥의 절환회수(s)이다.

마지막출력은 사용자에 대한 CPU 리용퍼센트(us), 체계의 CPU 리용퍼센트(sy), 놀고 있는 CPU 퍼센트(id)이다. 이것은 iostat 출력으로서는 완전하지 않으므로 nice 등록사항도 보아야 한다.

만일 체계가 강한 I/O 작업부하를 가지고 수행하고 있다면 수행할수 있는 프로쎄스들(r)에서 능동성, 봉쇄된 프로쎄스들(b), 수행 혹은 교환되는 프로쎄스(w)들이 얼마나 많은가를 보아야 할것이다. 만일 수행할수 있는 프로쎄스들이 많거나 프로쎄스들이 교환되고 있다면 체계는 대체로 I/O 병목을 가진다는것을 의미한다.

netstat 에 의한 망의 통계작성

netstat 는 망의 통계자료에 관한 정보를 제공한다. 망에서 **대역너비**(Bandwidth)가 일부 망의 수행에 CPU 와 기억만큼 영향을 많이 줄수 있기때문에 망통신량의 수준에 대한 개념을 알고 있어야 할것이다.

여기서는 망의 통계자료를 얻기 위하여 netstat 의 두 형식을 리용하고 있다. 한가지 형식은 netstat -i 이고 자동적으로 모형화되는 **대면부**(Interface)의 상태를 보여 준다. 비록 netstat -i 는 망에 가입, 그 가입의 이름, 다 같은 1 차 LAN 대면부의 좋은 개요를 주고 있지만 유용한 통계적정보는 주지 못하고 있다.

다음 실례는 netstat -i 의 출력정보를 보여 준다.

```
# netstat -i
Name   Mtu    Network   Address      Ipkts    Ierrs   Opkts    Oerrs   Col
lan0   1497   151.150   a4410.e.h.c  242194   120     107665   23      19884
```

netstat 는 간단한 출력을 제공한다. 다른 방법으로 출력하면 netstat 는 유용한 많은 정보를 제공한다. 아래에서는 웃 실례에 포함되여 있는 9 개 마당에 대한 해설을 하기로 한다. 즉

Name	자기의 망대면부의 이름(Name)이 있고 이 실례에서는 lan0 이다.
Mtu	최대전송단위이고 대면부카드에 의하여 보내지는 파케트(Packet)단위로 최대크기를 준다.
Network	대면부카드가 접속된 LAN 의 망주소이다. (151.150)
Address	자기가 가입된 체계의 호스트이름이다. 망이 /etc/hosts 리용으로 모형화되였다면 그 이름은 /etc/hosts 파일에서 나타나는것과 같은 파일체계의 기호적이름이다.

아래에서는 통계적정보를 주었다. 리용하고 있는 체계에 따라서 혹은 OS 의 방안에 따라서 이 지령의 일부 정보는 없을수도 있다.

Ipkts	대면부카드로 접수될 파케트들의 개수이며 실례에서는 lan0 이다.
Ierrs	대면부카드로 들어 온 파케트들에서 발견된 오유의 개수이다.
Opkts	대면부카드로 전송될 파케트들의 개수이다.
Oerrs	대면부카드로 파케트를 전송할 때에 발견된 오유의 개수이다.
Col	파케트의 통신조종으로 일어난 충돌의 개수이다.

netstat 는 마디점이 마지막에 투입된후에 루적한 자료를 제공한다. 그러므로 자료가 축적되는데는 많은 시간을 소비해야 할것이다. 만일 유용한 통계적정보를 보는데 관심을 가진다면 netstat 는 다른 선택항목들을 리용할수 있다. 통계자료의 정보표시에 시간구간도 규정할수 있다. 체계가 마지막에 투입된후의 모든 자료를 볼수 있기때문에 첫 등록항목의 정보를 보통 무시하고 있다. 이것은 체계가 놀고 있을 때의 자료가 **부차시간**(Non-Prime Hours)을 포함한다는것을 의미한다. 여기서는 체계가 큰 부하로 작업하고 있을 때 자료를 보도록 제공한다. 다음 실례들은 solaris, HP-UX, AIX 의 LAN 대면부규정으로 netstat -i 의 수행을 보여 주고 있다. 이 출력들은 비록 망대면부의 이름은 UNIX 의 방안에 따라 다르지만 대체로 동일하다. 지령은 5 개 부분의 시간구간에서 수행한다. Linux 방안에서 netstat 지령은 시간구간규정을 할수 없다.

Solaris 실례:

\# **netstat -I le0 5**

input	le0	output			input	(Total)	output		
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
116817990	0	3299582	11899	1653100	116993185	0	3474777	11899	1653100
185	0	3	0	0	185	0	3	0	0
273	0	8	0	0	273	0	8	0	0
153	0	3	0	0	153	0	3	0	0
154	0	3	0	0	154	0	3	0	0
126	0	3	0	0	126	0	3	0	0
378	0	2	0	0	378	0	2	0	0
399	0	4	0	0	399	0	4	0	0
286	0	2	0	0	286	0	2	0	0

HP-UX 실례 (10.x):

netetat -I lan0 5

(lano)-> input		output			(Total)-> input		output		
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
269841735	27	256627585	1	5092223	281472199	27	268258048	1	5092223
1602	0	1238	0	49	1673	0	1309	0	49
1223	0	1048	0	25	1235	0	1060	0	25
1516	0	1151	0	42	1560	0	1195	0	42
1553	0	1188	0	17	1565	0	1200	0	17
2539	0	2180	0	44	2628	0	2269	0	44
3000	0	2193	0	228	3000	0	2193	0	228
2959	0	2213	0	118	3003	0	2257	0	118
2423	0	1981	0	75	2435	0	1993	0	75

AIX 실례:

netstat -I en0 5

input	(en0)	output			input	(Total)	output		
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
46333531	0	1785025	0	0	47426087	0	2913405	0	0
203	0	1	0	0	204	0	2	0	0
298	0	1	0	0	298	0	1	0	0
293	0	1	0	0	304	0	12	0	0
191	0	1	0	0	191	0	1	0	0
150	0	2	0	0	151	0	3	0	0
207	0	3	0	0	218	0	15	0	0
162	0	3	0	0	162	0	4	0	0
120	0	2	0	0	120	0	2	0	0

이 실례에서는 출력의 오른쪽에서 총 합을 포함시키여 LAN 대면부에서 수행되는 작업에 대한 다중출력을 제시하였다. 이미 앞에서 언급한바와 같이 긴 시간주기에 대한 정보를 포함하기때문에 첫 출력을 무시하여도 된다. 이것은 망에서 작업을 하지 않고 있을때의 시간을 포함할수 있으므로 자료는 중요하지 않을수도 있다.

-I interface 는 정보들의 통계자료가 요구되는 망대면부를 규정한다. 실례에서 그 대면부는 -I 와 le0, lan0, en0 중의 하나이다. 5s 구간은 이 실례에서도 리용되였다.

netstat 통계적정보의 해석은 직관적이다. 충돌(Colls)마당에서 제시된 값은 전송된 파케트(Opkts)마당의 값보다 훨씬 작다는것을 확인할수 있을것이다. 충돌은 LAN 대면부로

부터 출력에서 나타나고 있다. LAN 대면부의 매 충돌한 체계의 성분들은 망을 느리게 한다. 어떤 충돌이 너무 많은가 하는데 대한 통보를 보자면 다른 선택항목들을 주어야 한다. 만일 충돌이 Opkts 의 5%보다 작다면 아마 좋은 모형일것이고 다른 일부 체계자원을 해석하는 시간을 줄이는 좋은 방법으로 된다. 만일 이 시간이 크다면 많은 자료를 공유하지 않는 망의 부분들로 망의 요구를 설정하는것과 같은 방법으로 망의 범위를 여러개로 토막화할수 있다.

만일 수신 및 송신하는 파케트의 개수(Ipkts 와 Opkts)를 줄인다면 총체적으로 더 적은 망통신량 및 충돌이 있게 된다. 망의 계획화 혹은 리용하는 체계의 질을 개선하는것으로 자기의 요구에 맞는 체계환경을 보장 받을수 있다. 항시적전송을 가지는 체계에서는 두개의 LAN 카드를 가질수 있다. 이런 체계들은 통신에 개별적 LAN 의 특징를 가지고 망에서 통신을 하는데 다른 체계의 수행에는 영향을 주지 않는다. 매 체계에서 한개의 LAN 대면부는 체계내부에서의 전송을 말아 수행한다. 이 대면부는 보통 봉사기로 역할을 하는 체계처럼 규정된 전송경로를 보장 받는다. 둘째 LAN 대면부는 큰 망에서 보통 체계가 의뢰기와 통신을 보장하는데 리용한다.

netstat 에 의하여 망경로화에 관계되는 정보를 얻을수도 있다. netstat 에 선택항목 -r 는 보통 알고 싶어하는 경로화표를 제시하여 주고 선택항목 -n 은 이름이 아니라 수로서 망주소인쇄에 리용할수 있다는것을 제시하여 준다. 다음의 실례들에서는 netstat 에 선택항목 -r 를 주어 수행시킨 결과(이것은 netstat 출력을 해설할 때 리용된다.)와 선택항목 -rn 을 주어 수행시킨 결과를 비교해 볼수 있을것이다.

$ netstat -r

Routing tables

Destination	Gateway	Flags	Refs	Use	Interface	Pmtu
hp700	localhost	UH	0	28	lo0	4608
default	routerl	UG	0	0	lan0	4608
128.185.61	systeml	U	347	28668	lan0	1500

$ netstat -rn

Routing tables

Destination	Gateway	Flags	Refs	Use	Interface	Prntu
127.0.0.1	127.0.0.1	UH	0	28	lo0	4608
default	128.185.61.1	UG	0	0	lan0	4608
128.185.61	128.185.61.2	U	347	28668	lan0	1500

netstat 에 의하여 제시된 일부 정보는 중간 등록항목인 통로에 대한 정보를 제공한다. 선택항목 -r 는 경로화에 대한 통보를 보여 주지만 이 지령에서 리용할수 있는 편리한 선택항목들은 많다. 이 출력에서 특별한 관심을 가지는것은 마당 Flags 인데 이것을 리용하

면 필요한 경로의 형태를 규정할수 있다. 여기서는 UNIX 방안에 따라 기발들이 달라 질수 있는데 기발들에 대한 해설은 이 장의 마감에 있는 편람에서 찾아 볼수 있을것이다.

1=U　　　국부적호스트인 **관문**(Gateway)을 통하여 망에로의 경로를 규정한다.
3=UG　　원격호스트인 관문을 통하여 망에로의 경로를 규정한다.
5=UH　　국부적호스트자체인 관문을 통하여 호스트에로의 경로를 규정한다.
7=UGH　 호스트인 원격관문을 통하여 호스트에로의 경로를 규정한다.

첫 행은 국부적호스트에 대한 정보이고 주소 127.0.0.1 에서 lo0 이라고 부르는 대면부를 고리로 련결한다. (netstat -rn 실례에서 이 주소를 볼수 있다.) UH 기발은 목표주소가 국부적호스트자신이라는것을 지적한다. 이 클라스 A 의 주소로써 TCP/IP 를 통하여 다른 콤퓨터와 통신을 할 때 어떤 호스트는 의뢰기 혹은 봉사기로도 될수 있다. 자료기입은 망에서 나가지 않는 대면부를 고리로 묶기 위한것이다. 그 자료기입은 단순히 거꾸로 고리를 통하여 실현된다.

둘째 행은 표준기정경로에 대한 정보이다. 만일 규정된 경로가 더는 찾을수 없으면 이 등록사항은 경로 1 에 파케트를 보낸다는것을 의미한다. 이 경우에 경로는 Flags 밑에 UG 를 가지고 있다. 일부 경로들은 U 에 의하여 모형화된다. 이 실례에서와 같이 다른 성분들은 UG 에 의하여 모형화된다. 실험에 의하면 U 혹은 UG 가 요구될 때마다 시험은 오유를 통하여 종결되며 끝난다는것을 알수 있었다. 만일 Flags 에서 U 이고 한 경로의 다른 쪽에서 체계가 통보보내기와 응답받기를 할수 없다면 UG 등록사항은 보통 문제를 제기한다.

셋째 행은 체계의 망대면부인 lan0 의 정보이다. 이것은 128.185.61 에 보내야 할 파케트에 대하여 이 망대면부를 리용해야 한다는것을 의미한다.

PS 에 의한 프로쎄스들의 검사

체계에서 수행하는 **프로쎄스**(Process)들에 대해서와 그것들이 어떻게 정리리용되는가에 대한 지식은 체계조직과 수행에서 중요한것들로 된다.

자기의 체계가 무엇을 하고 있는가 하는데 대한 대답을 찾으려면 ps -ef 를 리용해야 한다. 이 지령은 체계에서 수행하는 매개 프로쎄스에 대한 정보를 제공한다. 실례로 만일 NFS 가 수행되는가를 알려면 간단하게 ps -ef 로 알아 볼수 있다. 그러자면 NFS 데몬(daemon)을 보아야 한다. 비록 ps 는 체계에서 수행하고 있는 매 프로쎄스들에 대하여 말하고 있지만 체계자원의 리용정도에 대한 좋은 정보는 제공하지 않는다. ps 는 체계조직지령으로 대부분의 경우에 자주 사용된다. ps 에서는 리용할수 있는 몇개의 선택항목들이 있다. 여기서 표준적으로 e 와 f 를 리용하여 수행하는 매 프로쎄스들에 대한 정보(e)를 제공하고 또 이 정보를 완전히 목록으로 제공한다(f). ps 의 출력은 체계마다 거의 동일하다. 다음의 매개 실례는 각각 solaris, AIX, HP-UX 체계로부터 얻는것이다.

Solaris 실례:

```
martyp $ ps -ef
    UID   PID  PPID  C  STIME   TTY      TIME  CMD
   root     0     0  0  Feb 18  ?        0:01  sched
   root     1     0  0  Feb 18  ?        1:30  /etc/init-
   root     2     0  0  Feb 18  ?        0:02  pageout
   root     3     0  1  Feb 18  ?      613:44  fsflush
   root  3065  3059  0  Feb 22  ?        5:10  /usr/dt/bin/sdtperfmeter –f –H -r
   root    88     1  0  Feb 18  ?        0:01  /usr/sbin/in.routed -q
   root   478     1  0  Feb 18  ?        0:00  /usr/lib/saf/sac –t 300
   root    94     1  0  Feb 18  ?        2:50  /usr/sbin/rpcbind
   root   150     1  0  Feb 18  ?        6:03  /usr/sbin/syslogd
   root    96     1  0  Feb 18  ?        0:00  /usr/sbin/keyserv
   root   144     1  0  Feb 18  ?       50:37  /usr/lib/autofs/automountd
   root  1010     1  0  Apr 12  ?        0:00  /opt/perf/bin/midaemon
   root   106     1  0  Feb 18  ?        0:02  /usr/lib/netsvc/yp/ypbind -broadt
   root   156     1  0  Feb 18  ?        0:03  /usr/sbin/cron
   root   176     1  0  Feb 18  ?        0:00  /usr/lib/lpsched
   root   129     1  0  Feb 18  ?        0:00  /usr/lib/nfs/lockd
 daemon   130     1  0  Feb 18  ?        0:01  /usr/lib/nfs/statd
   root 14798     1  0  Mar 09  ?       31:10  /usr/sbin/nscd
   root   133     1  0  Feb 18  ?        0:10  /usr/sbin/inetd -s
   root   197     1  0  Feb 18  ?        0:00  /usr/lib/power/powerd
   root   196     1  0  Feb 18  ?        0:35  /etc/opt/licenses/lmgrd.ste -c /d
   root   213     1  0  Feb 18  ?     4903:09  /usr/sbin/vold
   root   199   196  0  Feb 18  ?        0:03  suntechd -T 4 –c /etc/optd
   root   219     1  0  Feb 18  ?        0:08  /usr/lib/sendmail    -bd     -ql5m
   root   209     1  0  Feb 18  ?        0:05  /usr/lib/utmpd
   root  2935   266  0  Feb 22  ?       48:08  /usr/openwin/bin/Xsun :0 -nobanna
   root 16795 16763  107:51:34 pts/4     0:00  ps-ef
   root  2963  2954  0  Feb 22  ?        0:17  /usr/openwin/bin/fbconsole
   root   479     1  0  Feb 18  console  0:00  /usr/lib/saf/ttymon –g –h –p sunc
   root 10976     1  0  Jun 01  ?        0:00  /opt/perf/bin/ttd
   root  7468     1  0  Feb 24  ?        0:13  /opt/perf/bin/pvalarmd
   root   266     1  0  Feb 18  ?        0:01  /usr/dt/bin/dtlogin   -daemon
 martyp 16763 16761  007:46:46 pts/4     0:01  -ksh
   root 10995     1  0  Jun 01  ?        0:01  /opt/perf/bin/perflbd
   root   484   478  0  Feb 18  ?        0:00  /usr/lib/saf/ttymon
```

```
    root    458      1    0  Feb  18   ?         20:06  /usr/lib/snmp/snmpdx  –y  –c  /etc/f
    root 16792   3059    007:50:37   ?          0:00  /usr/dt/bin/dtscreen  -mode    blank
    root    471      1    0  Feb  18   ?          0:07  /usr/lib/dmi/dmispd
    root    474      1    0  Feb  18   ?          0:00  /usr/lib/dmi/snmpXdmid  -s
    root    485    458    0  Feb  18   ?        739:44  mibiisa  –r  –p  32874
    root   2954   2936    0  Feb  22   ?          0:01  /bin/ksh     /usr/dt/bin/Xsession
    root   2936    266    0  Feb  22   ?          0:00  /usr/dt/bin/dtlogin    -daemon
    root   3061   3059    0  Feb  22   ?          1:32  dtwm
    root   3058      1    0  Feb  22   pts/2      0:01  /usr/dt/bin/ttsession
    root    712    133    0  Feb  18   ?          0:01  rpc.ttdbserverd
    root 11001 11000    0                          0:01  <defunct>
    root   2938      1    0  Feb  22   ?          0:00  /usr/openwin/bin/fbconsole  -d  :0
    root   2999   2954    0  Feb  22   pts/2      0:16  /usr/dt/bin/sdt-shell  -c unt
    root   3059   3002    0  Feb  22   pts/2    283:35  /usr/dt/bin/dtsession
    root   3063   3059    0  Feb  22   ?          0:03  /usr/dt/bin/dthelpview  -helpVolur
    root   3099   3062    0  Feb  22   ?          0:13  /usr/dt/bin/dtfile  –geometry  +700
    root 11000 10995    0   Jun  01   ?          0:02  /opt/perf/bin/agdbserver  –t  alar/
    root   3002   2999    0  Feb  22   pts/2      0:01  -ksh  –c unset  DT;  DISPLg
    root    730    133    0  Feb  18   ?          1:37  rpc.rstatd
    root   3062   3059    0  Feb  22   ?          2:17  /usr/dt/bin/dtfile  –geometry  +700
    root   3067      1    0  Feb  22   ?          0:00  /bin/ksh  /usr/dt/bin/sdtvolcheckm
    root   3000      1    0  Feb  22   ?          0:00  /usr/dt/bin/dsdm
    root   3078   3067    0  Feb  22   ?          0:00  /bin/cat  /tmp/.removable/notifyo
    root 10984      1    0   Jun  01   ?         12:42  /opt/perf/dce/bin/dced  -b
    root 16761    133    007:46:45   ?          0:00  in.telnetd
martyp  $
```

AIX 실례:

```
martyp  $ ps  -ef
     UID    PID  PPID    C  STIME    TTY       TIME  CMD
    root      1      0    0   Feb 24   -          5:07  /etc/init
    root   2208 15520    0   Feb 24   -          8:21  dtwm
    root   2664      1    0   Feb 24   -          0:00  /usr/dt/bin/dtlogin  -daemon
    root   2882      1    0   Feb 24   -        158:41  /usr/sbin/syncd  60
    root   3376   2664    5   Feb 24   -       3598:41  /usr/lpp/Xii/bin/X  –D  /usr/lib/
    root   3624   2664    0   Feb 24   -          0:00  dtlogin  <:0> -daemon
    root   3950      1    6   Feb 24   -       5550:30  /usr/lpp/perf/bin/llbd
    root   4144      1    0   Feb 24   -          0:00  /usr/lpp/perf/bin/midaemon
```

```
root  4490     1   0  Feb 24  -        0:48  /usr/lpp/perf/bin/perflbd
root  4906     1   0  Feb 24  -        0:00  /usr/lib/errdemon
root  5172     1   0  Feb 24  -        0:00  /usr/sbin/srcmstr
root  5724  5172   0  Feb 24  -        9:54  /usr/sbin/syslogd
root  6242  5172   0  Feb 24  -        0:00  /usr/sbin/biod  6
root  6450  5172   0  Feb 24  -        0:02  sendmail:  accepting  connections
root  6710  5172   0  Feb 24  -        7:34  /usr/sbin/portmap
root  6966  5172   0  Feb 24  -        0:23  /usr/sbin/inetd
root  7224  5172   0  Feb 24  -        1:09  /usr/sbin/timed  -S
root  7482  5172   0  Feb 24  -       11:55  /usr/sbin/snmpd
root  8000     1   0  Feb 24  -        9:17  ovspmd
root  8516  8782   0  Feb 24  -        0:00  netfmt  -CF
root  8782     1   0  Feb 24  -        0:00  /usr/OV/bin/ntl  reader  0  1  1  1
root  9036  8000   0  Feb 24  -       10:09  ovwdb  –0  -n5000
root  9288  8000   0  Feb 24  -        0:44  pmd  –Au  –At  –Mu  –Mt  -m
root  9546  8000   0  Feb 24  -       20:05  trapgend  -f
root  9804  8000   0  Feb 24  -        0:28  trapd
root 10062  8000   0  Feb 24  -        0:47  orsd
root 10320  8000   0  Feb 24  -        0:33  ovesmd
root 10578  8000   0  Feb 24  -        0:30  ovelmd
root 10836  8000   0  Feb 24  -       13:12  ovtopmd  -0
root 11094  8000   0  Feb 24  -       17:50  netmon  -P
root 11352  8000   0  Feb 24  -        0:02  snmpCollect
root 11954     1   0  Feb 24  -        1:22  /usr/sbin/cron
root 12140  5172   0  Feb 24  -        0:01  /usr/lib/netsvc/yp/ypbind
root 12394  5172   0  Feb 24  -        1:39  /usr/sbin/rpc.mountd
root 12652  5172   0  Feb 24  -        0:29  /usr/sbin/nfsd  8
root 12908  5172   0  Feb 24  -        0:00  /usr/sbin/rpc.statd
root 13166  5172   0  Feb 24  -        0:29  /usr/sbin/rpc.lockd
root 13428     1   0  Feb 24  -        0:00  /usr/sbin/uprintfd
root 14190  5172   0  Feb 24  -       72:59  /usr/sbin/automountd
root 14452  5172   0  Feb 24  -        0:17  /usr/sbin/qdaemon
root 14714  5172   0  Feb 24  -        0:00  /usr/sbin/writesrv
root 14992     1   0  Feb 24  -      252:26  /usr/lpp/perf/bin/scopeux
root 15520  3624   1  Feb 24  -       15:29  /usr/dt/bin/dtsession
root 15742     1   0  Feb 24  -        0:00  /usr/lpp/diagnostics/bin/diagd
root 15998     1   0  Feb 24  1ft0     0:00  /usr/sbin/getty  /dev/console
root 16304 18892   0  Feb 24  pts/0    0:00  /bin/ksh
root 16774     1   0   Feb24  -        0:00  /usr/lpp/perf/bin/ttd
```

```
     root 17092   4490     0  Feb  24   -            68:54   /usr/lpp/perf/bin/rep_server  -t
     root 17370 19186     3                           0:00   <defunct>
     root 17630 15520     0   Mar 25   -              0:00   /usr/dt/bin/dtexec  –open  0  -ttp
     root 17898 15520     0   Mar 20   -              0:00   /usr/dt/bin/dtexec  –open  0  -ttp
     root 18118 19888     0   Feb 24   pts/1          0:00   /bin/ksh
     root 18366   6966     0   Feb 24   -             0:00   rpc.ttdbserver  100083  1
     root 18446 15520     0   Mar 15   -              0:00   /usr/dt/bin/dtexec  –open  0  -ttp
     root 18892 15520     0   Feb 24   -              3:46   /usr/dt/bin/dtterm
     root 19186 16304     0   Feb 24   pts/0          0:01   /usr/1pp/X11/bin/msmit
     root 19450         1     0   Feb 24   -         26:53   /usr/dt/bin/ttsession -s
     root 19684   2208     0   Feb 24   -             0:00   /usr/dt/bin/dtexec  –open  0  -ttp
     root 19888 19684     0   Feb 24   -              0:00   /usr/dt/bin/dtterm
     root 20104 15520     0   Feb 27   -              0:00   /usr/dt/bin/dtexec  -open  0  -ttp
     root 20248 20104     0   Feb 27   -              0:03   /usr/dt/bin/dtscreen
     root 20542 29708     0  May 14   -               0:03   /usr/dt/bin/dtscreen
     root 20912 26306     0   Apr 05   -              0:03   /usr/dt/bin/dtscreen
     root 33558         1     0  May 18   -           3:28   /usr/atria/etc/lockmgr  -a  /var/
     root 33834   6966     307:55:49   -              0:00   telnetd
     root 34072         1     0 May  18   -           0:00   /usr/atria/etc/albd-server
  martyp 36296 36608   1307:56:07   pts/2             0:00   ps-ef
  martyp 36608 33834     107:55:50   pts/2            0:00   -kSh
     root 37220 15520     0 May  28   -               0:00   /usr/dt/bin/dtexec  –open  0  -ttp
martyp  $
```

HP-UX 실례 (부분적목록화):

```
martyp  $  ps  -ef
UID          PID      PPID     C        STIME  TTY     TIME     COMMAND
root           0        0        0        Mar  9   ?       107:28   swapper
root           1        0        0        Mar  9   ?       2:27     init
root           2        0        0        Mar  9   ?       14:13    vhand
root           3        0        0        Mar  9   ?       114:55   statdaemon
root           4        0        0        Mar  9   ?       5:57     unhashdaemon
root           7        0        0        Mar  9   ?       154:33   ttisr
root          70        0        0        Mar  9   ?       0:01     lvmkd
root          71        0        0        Mar  9   ?       0:01     lvmkd
root          72        0        0        Mar  9   ?       0:01     lvmkd
root          13        0        0        Mar  9   ?       9:54     vx_sched_thread
root          14        0        0        Mar  9   ?       1:54     vx_iflush_thread
```

root	15	0	0	Mar 9	?	2:06	vx_ifree_thread
root	16	0	0	Mar 9	?	2:27	vx_inactive_cache_thread
root	17	0	0	Mar 9	?	0:40	vx_delxwri_thread
root	18	0	0	Mar 9	?	0:33	vx_logflush_thread
root	19	0	0	Mar 9	?	0:07	vx_attrsync_thread

⟆

root	69	0	0	Mar 9	?	0:09	vx_inactive_thread
root	73	0	0	Mar 9	?	0:01	lvmkd
root	74	0	19	Mar 9	?	3605:29	netisr
root	75	0	0	Mar 9	?	0:18	netisr
root	76	0	0	Mar 9	?	0:17	netisr
root	77	0	0	Mar 9	?	0:14	netisr
root	78	0	0	Mar 9	?	0:48	nvsisr
root	79	0	0	Mar 9	?	0:00	supsched
root	80	0	0	Mar 9	?	0:00	smpsched
root	81	0	0	Mar 9	?	0:00	smpsched
root	82	0	0	Mar 9	?	0:00	sblksched
root	83	0	0	Mar 9	?	0:00	sblksched
root	84	0	0	Mar 9	?	0:00	strmem
root	85	0	0	Mar 9	?	0:00	strweld
root	3730	1	0	16:39:22	console	0:00	/usr/sbin/getty console console
root	404	1	0	Mar 9	?	3:57	/usr/sbin/swagentd
oracle	919	1	0	15:23:23	?	0:00	oraclegprd（LOCAL=NO）
root	289	1	2	Mar 9	?	78:34	/usr/sbin/syncer
root	426	1	0	Mar 9	?	0:10	/usr/sbin/syslogd -D
root	576	1	0	Mar 9	?	0:00	/usr/sbin/portmap
root	429	1	0	Mar 9	?	0:00	/usr/sbin/ptydaemon
root	590	1	0	Mar 9	?	0:00	/usr/sbin/biod 4
root	442	1	0	Mar 9	?	0:00	/usr/lbin/nktl daemon 000001-2
oracle	8145	1	0	12:02:48	?	0:00	oraclegprd（LOCAL=NO）
root	591	1	0	Mar 9	?	0:00	/usr/sbin/biod 4
root	589	1	0	Mar 9	?	0:00	/usr/sbin/biod 4
root	592	1	0	Mar 9	?	0:00	/usr/sbin/biod 4
root	604	1	0	Mar 9	?	0:00	/usr/sbin/rpc.lockd
root	598	1	0	Mar 9	?	0:00	/usr/sbin/rpc.statd
root	610	1	0	Mar 9	?	0:16	/usr/sbin/automount-f/etc/auto master
root	638	1	0	Mar 9	?	0:06	sendmail: accepting connections

```
root      618   1    0  Mar 9    ?   0:02   /usr/sbin/inetd
root      645   1    0  Mar 9    ?   5:01   /usr/sbin/snmpdm
root      661   1    0  Mar 9    ?   11:28  /usr/sbin/fddisubagtd
root      711   1    0  Mar 9    ?   30:59  /opt/dce/sbin/rpcd
root      720   1    0  Mar 9    ?   0:00   /usr/sbin/vtdaemon
root      867   777  1  Mar 9    ?   0:00   <defunct>
lp        733   1    0  Mar 9    ?   0:00   /usr/sbin/lpsched
root      777   1    0  Mar 9    ?   8:55   DIAGMON
root      742   1    0  Mar 9    ?   0:15   /usr/sbin/cron
oracle    7880  1    0  11:43:47 ?   0:00   oraclegprd（LOCAL=NO）
root      842   1    0  Mar 9    ?   0:00   /usr/vue/bin/vuelogin
oracle    5625  1    0  07:00:14 ?   0:01   ora_smon_gprd
root      781   1    0  Mar 9    ?   0:00   /usr/sbin/envd
root      833   777  0  Mar 9    ?   0:00 DEMLOG   DEMLOG;DEMLOG;0;
0;
root      813   1    0  Mar 9    ?   0:00   /usr/sbin/nfsd  4
root      807   1    0  Mar 9    ?   0:00   /usr/sbin/rpc.mountd
root      815   813  0  Mar 9    ?   0:00   /usr/sbin/nfsd  4
root      817   813  0  Mar 9    ?   0:00   /usr/sbin/nfsd  4
root      835   777  0  Mar 9    ?   0:13   PSMONPSMON;PSMON;0;0;
```

여기서 머리부에 대하여 간단한 해설을 하여 두자.

UID　　　　프로쎄스소유자의 사용자 ID

PID　　　　프로쎄스 ID(프로쎄스를 끝내기 위하여 이 수를 리용할수 있다.)

PPID　　　　선조프로쎄스의 프로쎄스 ID

C　　　　처리장치리용. 다중처리장치체계에서 이 수는 100%를 넘어선다는것을 알수 있다. 그것은 처리장치를 100%까지 리용한다고 가상적으로 볼수 있다. 두개의 처리장치의 리용은 200%로 될수 있다. 이것은 UNIX 방안에 따라 변한다.

STIME　　　　프로쎄스들의 시작시간

TTY　　　　프로쎄스를 조종할 말단장치

TIME　　　　프로쎄스의 루적된 수행시간

COMMAND　　　　지령이름과 인수들

ps 는 체계에서 수행하는 프로쎄스들을 빨리 개괄하여 준다. 더 상세한 통보를 보려면 선택항목 l 을 포함시킬수 있다. 다음 실례에 보여 주는것처럼 많은 유용한 추가적정보를 포함시킬수도 있다.

martyp $ **ps -efl**

```
 F S    UID   PID  PPID C PRI NI     ADDR   SZ    WCHAN  STIME     TTY D
19 T   root     0     0 0   0 SY f026f7f0    0            Feb 18   ?   d
 8 S   root     1     0 0  41 20 f5b90808  175 f5b90a30   Feb 18   ?   -
19 S   root     2     0 0   0 SY f5b90108    0 f0283fd0   Feb 18   ?   t
19 S   root     3     0 0   0 SY f5b8fa08    0 f0287a44   Feb 18   ?   6h
 8 S   root  3065  3059 0  40 20 f626d040 1639 f62aab96   Feb 22   ?   c
 8 S   root    88     1 0  40 20 f5b8d708  377 f5b59df6   Feb 18   ?   q
 8 S   root   478     1 0  41 20 f5b8ec08  388 f5b5lbb8   Feb 18   ?   0
 8 S   root    94     1 0  41 20 f5b8d008  527 f5b59e46   Feb 18   ?   d
 8 S   root   150     1 0  41 20 f5da1a10  808 f5b59806   Feb 18   ?   d
 8 S   root    96     1 0  67 20 f5da2810  535 f5b59ad6   Feb 18   ?   v
 8 S   root   144     1 0  41 20 f5da0c10 2694 ef69f6lc   Feb 18   ?   5d
 8 S   root  1010     1 0   0 RT f6lda330  496 f5dbeclc   Apr 12   ?   n
 8 S   root   106     1 0  41 20 f5da1310  485 f5b59e96   Feb 18   ?   s
 8 S   root   156     1 0  51 20 f5b8de08  446 f5b5lebB   Feb 18   ?   n
 8 S   root   176     1 0  53 20 f5da2110  740 f5b59036   Feb 18   ?   d
 8 S   root   129     1 0  56 20 f5d9fe10  447 f5b59cb6   Feb 18   ?   d
 8 S daemon   130     1 0  41 20 f5d9f710  564 f5b59b76   Feb 18   ?   d
 8 S   root 14798     1 0  45 20 f5b8e508  616 f5b8e730   Mar 09   ?   3d
 8 S   root   133     1 0  51 20 f5e18818  507 f5b59c66   Feb 18   ?   s
 8 S   root   197     1 0  63 20 f5e15e18  284 f5e16040   Feb 18   ?   d
 8 S   root   196     1 0  41 20 f5da0510  429 f5c68f8e   Feb 18   ?   c
 8 S   root   213     1 0  41 20 f5e16518  586 f5c68b2e   Feb 18   ?   4d
 8 S   root   199   196 0  41 20 f5el6clB  451 f5b59f86   Feb 18   ?   i
 8 S   root   219     1 0  41 20 f5el7318  658 f5b59d06   Feb 18   ?   m
 8 S   root   209     1 0  41 20 f5e18118  234 f5c68e4e   Feb 18   ?   d
 8 S   root  2935   266 0  40 20 f6ldb130 2473 f62aaa56   Feb 22   ?   4
 8 S   root 16800  3059 1  81 30 f626f340 1466 f6lb345e 07:59:40  ?   k
 8 S   root  2963  2954 0  40 20 f5f52028  513 f6lb3l3e   Feb 22   ?   e
 8 S   root   479     1 0  55 20 f5ee7120  407 f5fde2c6   Feb 18 consoleg
 8 S   root 10976     1 0  65 20 f5f55828  478 f5c6853e   Jun 01   ?   d
 8 S   root  7468     1 0  46 20 f621da38 2851    8306c   Feb 24   ?   d
 8 S   root   266     1 0  41 20 f5ee5520 1601 f5c6858e   Feb 18   ?   n
 8 S martyp 16763 16761 0  51 20 f6270140  429 f62701ac 07:46:46 pts/4 h
 8 S   root 10995     1 0  41 20 f5b8f308 2350 f5fde5e6   Jun 01   ?   d
 8 S   root   484   478 0  41 20 f5ee4e20  408 f5ee5048   Feb 18   ?   n
 8 S   root   458     1 0  41 20 f5f54a28  504 f5fde906   Feb 18   ?   2m
 8 O   root 16802 16763 1  61 20 f5ee7820  220          08:00:05 pts/4 1
```

```
  8  S  root   471     1 0  41  20  f5f53c28   658  f5fde726  Feb 18  ?      d
  8  S  root   474     1 0  51  20  f5f53528   804  f61a58b6  Feb 18  ?      g
  8  S  root   485   458 0  40  20  f5f52e28   734  f607ecde  Feb 18  ?      74
  8  S  root  2954  2936 0  40  20  f626e540   433  f626e5ac  Feb 22  ?      n
  8  S  root  2936   266 0  66  20  fSee4720  1637  f5ee478c  Feb 22  ?      n
  8  S  root  3061  3059 0  40  20  f5el7a18  2041  f6lb359e  Feb 22  ?      m
  8  S  root  3058     1 0  40  20  f6ldaa30  1067  f62aadc6  Feb 22  pts/2  n
  8  S  root   712   133 0  41  20  f6ld8e30   798  f6lb390e  Feb 18  ?      d
  8  Z  root 11001 11000 0   0                                                >
  8  S  root  2938     1 0  60  20  f5ee6320   513  f6olbfb6  Feb 22  ?      0
  8  S  root  2999  2954 0  40  20  f621e138  1450  f6lb33be  Feb 22  pts/2  t
  8  S  root  3059  3002 1  51  20  f626de40  4010  f62aafa6  Feb 22  pts/2  2n
  8  S  root  3063  3059 0  50  20  f621e838  1952  f62aa556  Feb 22  ?
  8  S  root  3099  3062 0  40  20  f5f52728  2275  f60a1d18  Feb 22  ?      0
  8  S  root 11000 10995 0  48  20  f626d740  2312     55694  Jun 01  ?      e
  8  S  root  3002  2999 0  43  20  f6ld8730   427  f6ldB79c  Feb 22  pts/2  =
  8  S  root   730   133 0  40  20  f61d9530   422  f62aa9b6  Feb 18  ?      d
  8  S  root  3062  3059 0  61  20  f62lb738  2275  f62aa506  Feb 22  ?      0
  8  S  root  3067     1 0  40  20  f5ee5c20   424  f5ee5c8c  Feb 22  ?      d
  8  S  root  3000     1 0  40  20  f6ld8030   518  f62aa8c6  Feb 22  ?      m
  8  S  root  3078  3067 0  40  20  f6ld9c30   211  f5b5l2b8  Feb 22  ?      0
  8  S  root 10984     1 0  41  20  f5f54328  2484  eee46e84  Jun 01  ?      lb
  8  S  root 16761   133 0  44  20  f5ee4020   411  f5c6894e  07:46:45 ?     d
martyp $
```

이 실례에서 첫 렬은 기발 F이다. F는 내부기억에서 체계의 프로쎄스와 주어 진 프로쎄스가 언제 교환되였는가에 대한 8 진통보를 제공한다. 8 진수값은 때때로 체계에 따라서 변하는데 기발의 8 진값을 보기 위하여서는 자기의 체계에 대한 편람을 참고해야 한다.

S 는 상태를 의미한다. 실례에서 보여 준 대부분 프로쎄스에 대해서 S 로 표시되여 있는데 상태들은 활동정지, 대기, 수행, 재개, 종결상태 등으로 될수 있다. 또한 일부 값들은 체계마다 다를수 있으므로 편람들을 보아야 한다.

이 출력결과에서 몇개의 유용한 정보들은 다음과 같다.

NI	좋은 값
ADRR	프로쎄스들의 기억주소
SZ	물리적페지들에서 프로쎄스의 크기
WCHAN	프로쎄스가 대기하고 있는 사건

프로쎄스의 끝내기

만일 ps 지령을 출력하고 프로쎄스들가운데서 한 프로쎄스가 방해되는 프로쎄스라는 것이 판정되거나 혹은 정지할것을 바라는 큰 일감이 시작되였다면 지령 kill 을 리용하여 끝낼수 있다. kill 은 소속한 임의의 프로쎄스를 끝내게 한다. 추가적으로 상급사용자는 체계의 임의의 대부분 프로쎄스들을 끝낼수 있다.

소속된 프로쎄스를 끝내려면 단순히 지령 kill 과 프로쎄스 ID(PID)를 주면 된다. 다음의 실례는 한 프로쎄스의 끝내기, 프로쎄스소멸에 대한 검사를 하여 martyp 에 소속된 모든 프로쎄스를 찾으려고 할 때의 지령 ps의 리용에 대하여 보여 준다.

```
martyp $ ps -ef | grep martyp
    martyp 19336 19334 0 05:24:32 pts/4 0:01 -ksh
    martyp 19426 19336 0 06:01:01 pts/4 0:00 grep martyp
    martyp 19424 19336 5 06:00:48 pts/4 0:01 find / -name .login
martyp $ kill 19424
martyp $ ps -ef | grep martyp
    martyp 19336 19334 0 05:24:32 pts/4 0:01 -ksh
    martyp 19428 19336 1 06:01:17 pts/4 0:00 grep martyp
[1] + Terminated            find / -name .login &
martyp $
```

실례는 martyp 에 소속된 19424 프로쎄스가 끝났다는것을 보여 준다. 여기서는 지령 ps 를 다시 주는것으로 필요한 프로쎄스를 끝낼수 있다. 지령 ps 에 선택항목 **-u** 를 주면 규정된 가입이름을 가진 프로쎄스들의 목록을 표시할수 있다.

끝낼것을 요구하는 개수의 모든 프로쎄스의 목록을 공백으로 구분하여 지령 kill 뒤에 주면 한 지령행으로도 여러개의 프로쎄스들을 끝낼수 있다.

상급사용자로 가입하였다면 프로쎄스들을 끝낼 때 특별한 주의를 해야 한다. 이것은 체계의 수행방법에 역효과를 준다면 프로쎄스들을 수동적으로 재기동하거나 체계를 재시동하여야 한다.

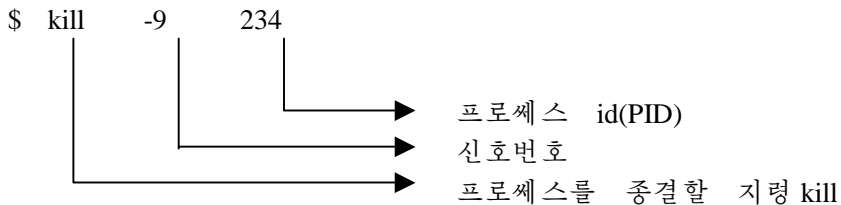
신호

지령 kill 과 프로쎄스의 개수를 줄 때 kill 과 관련된 **신호**(Signal)도 보낸다. 앞에서의 지령 kill 의 실례에서는 신호를 규정하지 않았지만 기정신호인 15(혹은 SIGTERM)가 리용되였다. 체계는 이 신호를 프로쎄스와 통신하는데 리용한다. 프로쎄스의 종결에 리용되는 15 인 신호는 쏘프트웨어의 종결신호인데 보통 사용자가 기동시킨 Find 와 같은 사용자프로쎄스를 종결시키기 위한 신호이다. 끝내기 곤난한 프로쎄스들은 SIGKILL 혹은 9 번 신호를 리용해야 한다. 이 신호는 프로쎄스를 직접 종결하게 한다. 여기서

SIGKILL 로 끝낸 프로쎄스들은 항상 원만히 종결할수 없기때문에 마지막재정돈만 리용하여야 한다. 쉘과 같은 프로쎄스들을 끝내려면 때때로 SIGKILL 을 리용해야 한다.

신호이름 혹은 신호번호인 수값을 리용할수도 있다. 이 신호번호는 체계마다 좀 다르므로 자기의 체계에서 신호목록을 보려면 신호의 편람의 다섯째 부분을 보면 된다. 대부분 자주 리용될 신호번호와 대응하는 신호이름들의 목록은 다음과 같다.

신호번호	신호이름
1	SIGHUP
2	SIGINT
3	SIGQUIT
9	SIGKILL
15	SIGTERM
24	SIGSTOP

SIGKILL 로 id 가 234 인 프로쎄스를 끝내려면 다음 지령을 줄수 있다.

showmount 에 의한 원격장비의 제시

showmount 는 국부적파일체계를 설치한 모든 원격체계(의뢰기)를 보는데 리용된다. showmount 는 대부분 NFS 로 의뢰기에 흔히 설치되는 파일체계를 결정하는데 편리할것이다. showmount 의 출력은 의뢰기에 설치된 호스트이름과 등록부의 목록이므로 특별히 쉽게 읽을수 있다.

NFS 봉사기들은 흔히 원래 봉사에 계획하지 않은 많은 NFS 의뢰기에 대한 봉사를 거절한다. 이런 상태는 추가적망의 대역너비만큼 NFS 봉사기에서 추가적 UNIX 체계자원들을 리용하면서 끝낸다. NFS 봉사기로부터 NFS 의뢰기에 전송된 임의의 자료는 망의 대역너비를 리용할것을 요구하고 있는데 만일 큰 파일이나 응용프로그람이 NFS 봉사기로부터 의뢰기에로 전송되고 있다면 일부 경우에 그 자료는 대역너비의 실제적인 총 량으로 될수 있다. 다음 실례는 체계로부터 취한 showmount 의 부분적출력을 보여 준다. showmount 는 이 장의 전반에서 리용하고 있는 HP-UX, AIX, Linux 체계에서 수행시켰으나 solaris 체계에서는 수행시키지 않았다.

```
# showmount -a
sys100.ct.mp.com:/applic
sys101.ct.mp.com:/applic
sys102.cal.mp.com:/applic
sys103.cal.mp.com:/applic
sys104.cal.mp.com:/applic
sys105.cal.mp.com:/applic
sys106.cal.mp.com:/applic
sys107.cal.mp.com:/applic
sys108.cal.mp.com:/applic
sys109.cal.mp.com:/applic
sys200.cal.mp.com:/usr/users
sys20l.cal.mp.com:/usr/users
sys202.cal.mp.com:/usr/users
sys203.cal.mp.com:/usr/users
sys204.cal.mp.com:/usr/users
sys205.cal.mp.com:/usr/users
sys206.cal.mp.com:/usr/users
# showmount -a
sys207.cal.rnp.com:/usr/users
sys208.cal.rnp.corn:/usr/users
sys209.cal.rnp.corn:/usr/users
```

다음의 세개의 선택항목들을 지령 showmount 에서 리용할수 있다.

-a 우에서 보여 준것처럼 "이름:등록부"형식으로 출력을 인쇄한다.
-d 의뢰기는 원격적으로 설치된 국부적인 모든 동록부들을 목록으로 표시한다.
-e 반출된 파일체계들의 목록을 인쇄한다.

다음 실례는 showmount -d 와 showmount -e 의 결과를 보여 준다.

```
# showmount -d
/applic
/usr/users
/usr/oracle
/usr/users/emp.data
/network/database
/network/users
/tmp/working
```

showmount -e
export list for server 101.cal.rnp.com
/applic
/usr/users
/cdrom

체계교환공간의 제시

만일 작업에 필요한 모든 정보를 기억시키는데 체계가 충분한 주기억을 가지고 있다면 정보의 페지들은 교환령역에 이동되거나 교환령역에서 프로쎄스전체들이 교환될것이다. 가장 최근에 리용된 페지들은 주기억에 있고 최근에 리용되지 않은 페지들은 처음으로 주기억밖으로 이동될것이다.

체계관리자는 체계의 교환령역의 정확한 총 량을 결정하는데 많은 시간을 소비한다. 불충분한 교환령역은 체계가 추가적프로쎄스들의 시작에 제한을 줄수도 있고 응용프로그람의 수행을 방해할수 있거나 추가적사용자들이 체계에 접근하지 못하도록 할수도 있다. 충분한 교환공간을 가지면 발생할수 있는 이런 문제들을 막을수 있다. 체계관리자는 보통 다음과 같은 많은 중요한 요인들을 고려하여 교환공간의 정확한 총 량을 결정하려고 한다.

① 얼마나 큰 교환령역이 응용프로그람들을 수행하는데 필요한가?

응용프로그람은 알선된 교환령역의 크기를 리용한다. 응용프로그람판매자들은 교환령역을 권고할 때 실제적으로 이 크기값이 요구된다. 총 가격을 줄이기 위한 한 방도로서 더 적은 기억과 CPU 를 리용하며 응용프로그람판매자들이 경쟁적으로 교환령역을 될수록 줄이려고 하고 있다.

② 얼마나 많은 응용프로그람들이 동시에 수행될수 있는가?

만일 여러개 응용프로그람들이 수행된다면 매 응용프로그람에 리용된 교환령역의 총 합은 동시적수행을 고려하여 결정해 주어야 한다. 만일 200MByte 교환을 리용하는 자료기지응용프로그람 100MByte 교환을 리용하는 개발수단이 있다면 최소로 300MByte 로 체계를 모형화해야 한다.

③ NFS 와 같은 보조기능에 대하여 실제적체계자원을 리용할것인가?

NFS 의 본질은 교환공간을 꽉 채우도록 요구하면서 대단히 큰 파일체계에 접근을 제공하는데 있다. 교환은 여러 지령들로써 다른 UNIX 방안에서 목록화하고 조작된다. 다음 실례는 solaris 체계에서 swap -l 로써 교환령역의 목록을 보여 준다.

```
# swap - l
swapfile              dev      swaplo  blocks  free
/dev/dsk/c0t3d0s1     32, 25   8       263080  209504
```

이 값들은 모두 512KByte 의 블로크의 개수이다. 이 경우에 자유블로크는 209504Byte 인데 이 값은 체계에 배치될 총 교환공간의 크기이다.

HP-UX 체계에서는 swapinfo 로 리용된 교환공간의 총 용량을 볼수 있다. 다음 실례는 swapinfo 의 출력을 보여 준다.

```
# swapinfo
              Kb       Kb       Kb     PCT   START/          Kb
TYPE       AVAIL     USED     FREE    USED    LIMIT     RESERVE   PRI  NAME
dev        49152    10532    38620     21%        0           -     1  /dev/vg00/lvol2
dev       868352    10888   759160      1%        0           -     1  /dev/vg00/1vo18
reserve        -   532360  -532360
memory    816360   469784   346576     58%
```

다음으로 swapinfo 가 무엇을 보여 주는가에 대한 간단한 개괄을 한다.

앞선 실례에서 TYPE 마당은 교환이 되는 장치에 대하여서는 dev, 예정된 페지공간에 대하여서는 reserve 혹은 memory 로 지적되였다.

Kb AVAIL	1024Byte 블로크로 리용될 총 교환공간이다. 이것은 리용되는 교환공간과 리용되지 않은 교환공간을 다 포함한다.
Kb USED	리용되는 1024Byte 블로크의 현재 개수이다.
Kb FREE	Kb AVAIL 과 Kb USED 사이의 차이값이다.
PCT USED	Kb USED 를 Kb AVAIL 로 나눈 값이다.
START/LIMIT	교환령역의 시작의 블로크이다.
Kb RESERVE	파일체계교환에 대하여서는 1024Byte 블로크의 수로 표시되고 교환장치에 대하여서는 −로 표시된다.
PRI	교환령역에 주어 진 우선권이다.
NAME	교환장치의 장치이름이다.

지령 swapinfo 은 선택항목들의 렬도 리용할수 있다. 여기서는 포함시킬수 있는 일부 선택항목들을 설명한다.

-m	KByte 블로크단위가 아니라 Mbyte 단위로 swapinfo 의 출력을 표시한다.
-d	장치교환령역에만 관계되는 정보를 인쇄한다.
-f	파일체계교환령역에 대한 통보만 인쇄한다.

sar : 체계능동성의 통보자

sar 는 체계에 대한 능동성에 대한 정보를 축적하기 위한 UNIX 의 다른 지령이다. sar 는 연장된 시간주기에 대한 자료를 수집할수도 있고 후에 그라프에 기초한 통보를 산생할수도 있다. UNIX 방안에 따라 sar 의 선택항목들은 류사하고 그 출력도 류사하다. 실례에서 리용한 Linux 체계는 sar 를 봉사하지 않지만 solaris, HP-UX, AIX 체계는 같은 선택항목들을 가지고 거의 동일하게 출력한다. 다음에 적용할수 있는 sar 에 편리한 일부 선택항목들을 산생된 통보의 실례와 함께 아래에서 제시한다.

sar -o　　o 로 규정된 파일에 자료를 보관한다. 보통 뒤에 파일이름과 상태의 시간구간과 상태의 개수도 입력해야 한다. 다음 실례는 60s 의 시간 간격으로 /tmp/sar.data 파일에 300 번 보관한 2 진자료를 보여 준다.

```
# sar -o/tmp/sar.data 60 300
```

/tmp/sar.data 에서 자료는 후에 파일로부터 출력될수 있다.

sar -f　　자료를 출력시킬 파일을 규정한다.

sar -u　　%usr, %sys, %wio, I/O 블로크를 기다리는 일부 프로쎄스에서 %idle, %idle 머리부에서 CPU 의 리용을 통보한다. 이 통보는 iostat 와 vmstat 의 CPU 의 통보와 류사하다. 다음 실례에서 보여 준 것처럼 CPU 정보를 얻기 위해서는 파일에 보관한 2 진자료를 출력해야 한다.

sar -u -f /tmp/sar.data

Header Information for your system

12:52:04	%usr	%sys	%wio	%idle
12:53:04	62	4	5	29
12:54:04	88	5	3	4
12:55:04	94	5	1	0
12:56:04	67	4	4	25
12:57:04	59	4	4	32
12:58:04	61	4	3	32
12:59:04	65	4	3	28
13:00:04	62	5	16	17
13:01:04	59	5	9	27

13:02:04	71	4	3	22
13:03:04	60	4	4	32
13:04:04	71	5	4	20
13:05:04	80	6	8	7
13:06:04	56	3	3	37
13:07:04	57	4	4	36
13:08:04	66	4	4	26
13:09:04	80	10	2	8
13:10:04	73	10	2	15
13:11:04	64	6	3	28
13:12:04	56	4	3	38
13:13:04	55	3	3	38
13:14:04	57	4	3	36
13:15:04	70	4	5	21
13:16:04	65	5	9	21
13:17:04	62	6	2	30
13:18:04	60	5	3	33
13:19:04	77	3	4	16
13:20:04	76	5	3	15
	⌇			
14:30:04	50	6	6	38
14:31:04	57	12	19	12
14:32:04	51	8	20	21
14:33:04	41	4	9	46
14:34:04	43	4	9	45
14:35:04	38	4	6	53
14:36:04	38	9	7	46
14:37:04	46	3	11	40
14:38:04	43	4	7	46
14:39:04	37	4	5	54
14:40:04	33	4	5	58
14:41:04	40	3	3	53
14:42:04	44	3	3	50
14:43:04	27	3	7	64
Average	57	5	8	30

sar -b　　고속완충기의 능동성을 통보한다. Oracle 과 같은 자료기지응용프로그람이 리용할 고속완충기의 유효성을 보기 위하여서는 이 선택항목을 리용할것을 권고한다. 다음 실례에 보여 준것처럼 CPU 정보를

얻기 위하여 파일에 보관된 2진자료를 출력한다.

sar -b -f /tmp/sar.data

Header information for your system

12:52:04	bread/s	lread/s	%rcache	bwrit/s	lwrit/s	%wcache	pread/s	pwrit/s
12:53:04	5	608	99	1	11	95	0	0
12:54:04	7	759	99	0	14	99	0	0
12:55:04	2	1733	100	4	24	83	0	0
12:56:04	1	836	100	1	18	96	0	0
12:57:04	0	623	100	2	21	92	0	0
12:58:04	0	779	100	1	16	96	0	0
12:59:04	0	1125	100	0	14	98	0	0
13:00:04	2	1144	100	9	89	89	0	0
13:01:04	10	898	99	11	76	86	0	0
13:02:04	0	1156	100	0	14	99	0	0
13:03:04	1	578	100	2	22	88	0	0
13:04:04	5	1251	100	0	12	99	0	0
13:05:04	3	1250	100	0	12	97	0	0
13:06:04	1	588	100	0	12	98	0	0
13:07:04	1	649	100	2	15	86	0	0
13:08:04	1	704	100	2	15	86	0	0
13:09:04	1	1068	100	0	18	100	0	0
13:10:04	0	737	100	1	44	99	0	0
13:11:04	0	735	100	1	13	95	0	0
13:12:04	0	589	100	1	15	93	0	0
13:13:04	0	573	100	0	16	99	0	0
13:14:04	1	756	100	1	16	91	0	0
13:15:04	1	1092	100	9	49	81	0	0
13:16:04	2	808	100	6	82	93	0	0
13:17:04	0	712	100	1	9	93	0	0
13:18:04	1	609	100	0	13	97	0	0
13:19:04	1	603	100	0	10	99	0	0
13:20:04	0	1127	100	0	14	98	0	0
		⌇						
14:30:04	2	542	100	1	22	94	0	0
14:31:04	10	852	99	12	137	92	0	0
14:32:04	2	730	100	10	190	95	0	0

14:33:04	4	568	99	2	26	91	0	0
14:34:04	4	603	99	1	13	91	0	0
14:35:04	1	458	100	1	13	89	0	0
14:36:04	13	640	98	1	24	98	0	0
14:37:04	21	882	98	1	18	95	0	0
14:38:04	7	954	99	0	19	98	0	0
14:39:04	3	620	100	1	11	94	0	0
14:40:04	3	480	99	2	15	85	0	0
14:41:04	1	507	100	0	9	98	0	0
14:42:04	1	1010	100	1	10	91	0	0
14:43:04	5	547	99	1	9	93	0	0
Average	3	782	100	3	37	91	0	0

sar -d　　디스크의 능동성을 통보한다. 장치이름, 장치가 실지 작업한 퍼센트, 장치에 대한 명백한 요청의 평균개수, 장치에 대한 초당 자료전송의 개수 등 다른 통보를 제시한다. 다음 실례에서 보여 준것처럼 CPU 정보를 얻기 위하여서는 파일에 보관된 2진자료를 출력한다.

sar -d -f /tmp/sar.data

Header information for your system

12:52:04	device	%busy	avque	r+w/s	blks/s	avwait	avserv
12:53:04	c0t6d0	0.95	1.41	1	10	16.76	17.28
	c5t4d0	100.00	1.03	20	320	8.36	18.90
	c4t5d1	10.77	0.50	13	214	5.02	18.44
	c5t4d2	0.38	0.50	0	3	4.61	18.81
12:54:04	c0t6d0	0.97	1.08	1	11	10.75	14.82
	c5t4d0	100.00	1.28	54	862	9.31	20.06
	c4t5d1	12.43	0.50	15	241	5.21	16.97
	c5t4d2	0.37	0.50	0	3	3.91	18.20
12:55:04	c0t6d0	1.77	1.42	1	22	13.32	14.16
	c5t4d0	100.00	0.79	26	421	8.33	16.00
	c4t5d1	14.47	0.51	17	270	5.30	13.48
	c5t4d2	0.72	0.50	0	7	4.82	15.69
12:56:04	c0t6d0	1.07	21.57	1	22	72.94	19.58
	c5t4d0	100.00	0.60	16	251	6.80	13.45
	c4t5d1	8.75	0.50	11	177	5.05	10.61

	c5t4d2	0.62	0.50	0	6	4.79	15.43
12:57:04	c0t6d0	0.78	1.16	1	9	13.53	14.91
	c5t4d0	100.00	0.66	15	237	7.60	13.69
	c4t5d1	9.48	0.54	13	210	5.39	13.33
	c5t4d2	0.87	0.50	1	10	4.86	14.09
12:58:04	c0t6d0	1.12	8.29	1	17	54.96	14.35
	c5t4d0	100.00	0.60	11	176	7.91	14.65
	c4t5d1	5.35	0.50	7	111	5.23	10.35
	c5t4d2	0.92	0.50	1	10	4.63	16.08
12:59:04	c0t6d0	0.67	1.53	1	8	18.03	16.05
	c5t4d0	99.98	0.54	11	174	7.69	14.09
	c4t5d1	3.97	0.50	5	83	4.82	9.54
	c5t4d2	1.05	0.50	1	11	4.69	16.29
13:00:04	c0t6d0	3.22	0.67	3	39	8.49	16.53
	c5t4d0	100.00	0.60	65	1032	8.46	14.83
	c4t5d1	21.62	0.50	31	504	5.30	8.94
	c5t4d2	6.77	0.50	5	78	4.86	14.09
13:01:04	c0t6d0	4.45	3.08	5	59	25.83	11.49
	c5t4d0	100.00	0.65	42	676	7.85	14.52
	c4t5d1	21.34	0.55	30	476	5.87	18.49
	c5t4d2	4.37	0.50	3	51	5.32	13.50
			⟆				
14:42:04	c0t6d0	0.53	0.83	0	7	12.21	16.33
	c5t4d0	100.00	0.56	7	107	6.99	14.65
	c4t5d1	6.38	0.50	7	113	4.97	15.18
	c5t4d2	0.15	0.50	0	2	4.53	16.50
14:43:04	c0t6d0	0.52	0.92	0	7	11.50	15.86
	c5t4d0	99.98	0.92	17	270	8.28	18.64
	c4t5d1	10.26	0.50	9	150	5.35	16.41
	c5t4d2	0.12	0.50	0	1	5.25	14.45
Average	c0t6d0	1.43	108.80	2	26	0.00	14.71
Average	c5t4d0	100.00	0.74	25	398	7.83	-10.31
Average	c4t5d1	19.11	0.51	25	399	5.26	-13.75
Average	c5t4d2	1.71	0.53	1	21	5.29	13.46

sar -q　　평균순서길이를 통보한다. 임의의 시간에 처리장치의 개수보다 큰 수행순서에서의 원소의 개수는 체계에서 문제를 야기할수 있다.

```
# sar -q -f /tmp/sar.data
```

Header information for your system

12:52:04	runq-sz	%runocc	swpq-sz	%swpocc
12:53:04	1.1	20	0.0	0
12:54:04	1.4	51	0.0	0
12:55:04	1.3	71	0.0	0
12:56:04	1.1	22	0.0	0
12:57:04	1.3	16	0.0	0
12:58:04	1.1	14	0.0	0
12:59:04	1.2	12	0.0	0
13:00:04	1.2	21	0.0	0
13:01:04	1.1	18	0.0	0
13:02:04	1.3	20	0.0	0
13:03:04	1.2	15	0.0	0
13:04:04	1.2	20	0.0	0
13:05:04	1.2	43	0.0	0
13:06:04	1.1	14	0.0	0
13:07:04	1.2	15	0.0	0
13:08:04	1.2	26	0.0	0
13:09:04	1.5	38	0.0	0
13:10:04	1.5	30	0.0	0
13:11:04	1.2	23	0.0	0
13:12:04	1.3	11	0.0	0
13:13:04	1.3	12	0.0	0
13:14:04	1.4	16	0.0	0
13:15:04	1.4	27	0.0	0
13:16:04	1.5	20	0.0	0
13:17:04	1.3	21	0.0	0
13:18:04	1.1	15	0.0	0
13:19:04	1.2	19	0.0	0
13:20:04	1.4	22	0.0	0
		⟆		
14:30:04	1.5	5	0.0	0
14:31:04	1.6	12	0.0	0
14:32:04	1.4	9	0.0	0
14:33:04	1.1	6	0.0	0
14:34:04	1.3	3	0.0	0

14:35:04	1.1	4	0.0	0
14:36:04	1.2	6	0.0	0
14:37:04	1.4	5	0.0	0
14:38:04	1.2	10	0.0	0
14:39:04	1.3	4	0.0	0
14:40:04	1.1	3	0.0	0
14:41:04	1.6	3	0.0	0
14:42:04	1.1	4	0.0	0
14:43:04	1.3	1	0.0	0
Average	1.3	17	1.2	0

sar -w 체계교환능동성을 통보한다.

\# **sar -w -f /tmp/sar.data**

Header information for your system

12:52:04	swpin/s	bswin/s	swpot/s	bswot/s	pswch/s
12:53:04	1.00	0.0	1.00	0.0	231
12:54:04	1.00	0.0	1.00	0.0	354
12:55:04	1.00	0.0	1.00	0.0	348
12:56:04	1.00	0.0	1.00	0.0	200
12:57:04	1.00	0.0	1.00	0.0	277
12:58:04	1.00	0.0	1.00	0.0	235
12:59:04	1.02	0.0	1.02	0.0	199
13:00:04	0.78	0.0	0.78	0.0	456
13:01:04	1.00	0.0	1.00	0.0	435
13:02:04	1.02	0.0	1.02	0.0	216
13:03:04	0.98	0.0	0.98	0.0	204
13:04:04	1.02	0.0	1.02	0.0	239
13:05:04	1.00	0.0	1.00	0.0	248
13:06:04	0.97	0.0	0.97	0.0	170
13:07:04	1.00	0.0	1.00	0.0	166
13:08:04	1.02	0.0	1.02	0.0	209
13:09:04	0.98	0.0	0.98	0.0	377
13:10:04	1.00	0.0	1.00	0.0	200
13:11:04	1.00	0.0	1.00	0.0	192

13:12:04	0.87	0.0	0.87	0.0	187
13:13:04	0.93	0.0	0.93	0.0	172
13:14:04	1.00	0.0	1.00	0.0	170
13:15:04	1.00	0.0	1.00	0.0	382
13:16:04	1.00	0.0	1.00	0.0	513
13:17:04	1.00	0.0	1.00	0.0	332
13:18:04	1.00	0.0	1.00	0.0	265
13:19:04	1.02	0.0	1.02	0.0	184
13:20:04	0.98	0.0	0.98	0.0	212
		⌇			
14:30:04	0.00	0.0	0.00	0.0	301
14:31:04	0.00	0.0	0.00	0.0	566
14:32:04	0.00	0.0	0.00	0.0	539
14:33:04	0.00	0.0	0.00	0.0	400
14:34:04	0.00	0.0	0.00	0.0	242
14:35:04	0.00	0.0	0.00	0.0	286
14:36:04	0.00	0.0	0.00	0.0	295
14:37:04	0.00	0.0	0.00	0.0	249
14:38:04	0.00	0.0	0.00	0.0	300
14:39:04	0.00	0.0	0.00	0.0	296
14:40:04	0.00	0.0	0.00	0.0	419
14:41:04	0.00	0.0	0.00	0.0	234
14:42:04	0.00	0.0	0.00	0.0	237
14:43:04	0.00	0.0	0.00	0.0	208
Average	0.70	0.0	0.70	0.0	346

지령을 해석하기 위한 timex

만일 소비된 시간, 사용자시간, 규정된 임의의 지령의 수행에 소비된 체계시간과 같은 더 많은 정보를 보려고 한다면 timex를 리용할수 있다.

사용자가 지령 timex을 줄 때 체계자원의 리용에 대한 평가를 주기때문에 이 지령은 편리하다. 다음의 두가지 실례는 소비된 CPU의 총 량에 대한 간단한 결과를 얻기 위하여 선택항목이 없이 timex를 주고 있다. 특히 둘째 실례는 solaris 체계에서 총 체계능동상태를 통보하기 위해 timex -s를 준 결과를 보여 주고 있다.

martyp $ **timex listing**

real 0.02

user 0.00

sys 0.02

```
martyp $ timex -s listing
real        0.02
user        0.00
sys         0.01

SunOS 5.7 Generic sun4m   08/21

07:48:30    %usr    %sys    %wio   %idle
07:48:31      32      68       0       0
07:48:30  bread/s  lread/s  %rcache  bwrit/s  lwrit/s  %wcache  pread/s  pwrit/s
07:48:31        0        0      100        0        0      100        0        0

Average         0        0      100        0        0      100        0        0

07:48:30   device   %busy   avque   r+w/s   blks/s   avwait   avserv
07:48:31   fd0          0     0.0       0        0      0.0      0.0
           nfsl         0     0.0       0        0      0.0      0.0
           nfs219       0     0.0       0        0      0.0      0.0
           sd1          0     0.0       0        0      0.0      0.0
           sd1，a       0     0.0       0        0      0.0      0.0
           sd1，b       0     0.0       0        0      0.0      0.0
           sd1，c       0     0.0       0        0      0.0      0.0
           sd1，g       0     0.0       0        0      0.0      0.0
           sd3          0     0.0       0        0      0.0      0.0
           sd3，a       0     0.0       0        0      0.0      0.0
           sd3，b       0     0.0       0        0      0.0      0.0
           sd3，c       0     0.0       0        0      0.0      0.0
           sd6          0     0.0       0        0      0.0      0.0

Average    fd0          0     0.0       0        0      0.0      0.0
           nfsl         0     0.0       0        0      0.0      0.0
           nfs2l9       0     0.0       0        0      0.0      0.0
           sd1          0     0.0       0        0      0.0      0.0
           sd1，a       0     0.0       0        0      0.0      0.0
           sd1，b       0     0.0       0        0      0.0      0.0
           sd1，c       0     0.0       0        0      0.0      0.0
           sd1，g       0     0.0       0        0      0.0      0.0
           sd3          0     0.0       0        0      0.0      0.0
           sd3，a       0     0.0       0        0      0.0      0.0
```

```
             sd3，b        0      0.0       0       0      0.0       0.0
             sd3，c        0      0.0       0       0      0.0       0.0
             sd6           0      0.0       0       0      0.0       0.0

07:48:30  rawch/s     canch/s   outch/s   rcvin/s   xmtin/s  mdmin/s
07:48:31        0           0       147         0         0        0

Average         0           0       147         0         0        0

07:48:30   scall/s    sread/s   swrit/s    fork/s    exec/s   rchar/s    wchar/s
07:48:31     2637           0        95     15.79     15.79         0    19216

Average      2637           0        95     15.79     15.79         0    19216

07:48:30  swpin/s     bswin/s   swpot/s   bswot/s   pswch/s
07:48:31     0.00         0.0      0.00       0.0       116

Average      0.00         0.0      0.00       0.0       116

07:48:30   iget/s     namei/s   dirbk/s
07:48:31        0         195       121

Average         0         195       121

07:48:30  runq-sz     %runocc   swpq-sz  %swpocc
07:48:31      2.0         526

Average       2.0         526

07:48:30  proc-sz    ov    inod-sz    ov   file-sz   ov    lock-sz
07:48:31   45/986     0   973/4508     0   357/357    0        0/0

07:48:30    msg/s  sema/s
07:48:31     0.00    0.00

Average      0.00    0.00

07:48:30   atch/s  pgin/s  ppgin/s   pflt/s    vflt/s  slock/s
07:48:31     0.00    0.00     0.00   505.26   1036.84    0.00
```

```
Average      0.00       0.00       0.00   505.26     1036.84       0.00

07:48:30       pgout/s      ppgout/s     pgfree/s    pgscan/s      %ufs_ipf
07:48:31          0.00          0.00        0.00        0.00          0.00

Average           0.00          0.00        0.00        0.00          0.00

07:48:30       freemem          freeswap
07:48:31         15084           1224421
Average          15084           1224421
07:48:30   sml_mem       alloc    fail      lg_mem         alloc    fail   ovsz_alloc     fail
07:48:31    2617344    1874368       0    17190912     10945416      0      3067904        0

Average      186953     133883       0     1227922       781815      0       219136        0
```

개선된 도형수행도구

UNIX 를 리용할 때에는 지령행에 의하여 작업하게 된다. UNIX 의 지령행의 종류는 늘어 나고 있지만 아직까지도 지령행에 기초하고 있다. 지령을 줄 때 많은것을 아는것이 필요하지만 본질적인 체계의 기능을 리용할 때 필요한 지령들을 배우기는 쉬울것이다.

최근의 UNIX 방안들에서는 도형적수행도구를 리용하기 위한 선택항목들을 많이 가지고 있다. 일부 체계는 기본적인 도형적수행도구들을 가지고 있지만 갱신된 수행해석을 하려고 할 때에는 일반적으로 갱신된 수행해석도구를 도입하면 된다. 아래 부분에서는 일부 수행도구들을 볼수 있다.

그림 14-1 은 이 장의 실례들에서 많이 리용하는 Red Hat Linux 체계의 세개 수행도구들을 보여 준다.

그림에서 세개의 수행도구들은 아래의 오른쪽부분에 xosview, 화면의 꼭대기에 체계모니터, 아래의 왼쪽부분에 top 이다. 체계모니터는 화면꼭대기에 CPU, Memory, Swap, LAN 리용의 총 량을 지적하는 띠도표를 표시한다. 그리고 체계에 있는 매 프로쎄스에 대한 표자료가 있다. 체계모니터는 도형적수행도구 top 를 위한 지령 gtop 로 발동된 top 의 도형적방안이다. xosview 는 밑의 오른쪽창문에 있는 체계능동성의 띠도표를 표시하는데 그것이 체계에 주는 부하는 작다. 이것은 X Windows 조작체계의 view 프로그람이며 그 이름은 xosview 이다. 이 도표에서 명백히 띠도표를 볼수 없는것은 그것이 색을 리용하는 응용프로그람이지만 우의 그림에서 흑백으로만 인쇄되였기때문이다. 그러나 띠도표는 콤퓨터화면에서 명백히 볼수 있다. 마지막의것은 때때로 리용되는 UNIX 체계의 도구가 밑의 왼쪽 Xterm 에서 수행되는 top 의 기호적방안이라는것을 보여 준다. top 는 많은 UNIX 방안들에도 있는데 그것은 많은 유용한 체계통보를 제시해 준다.

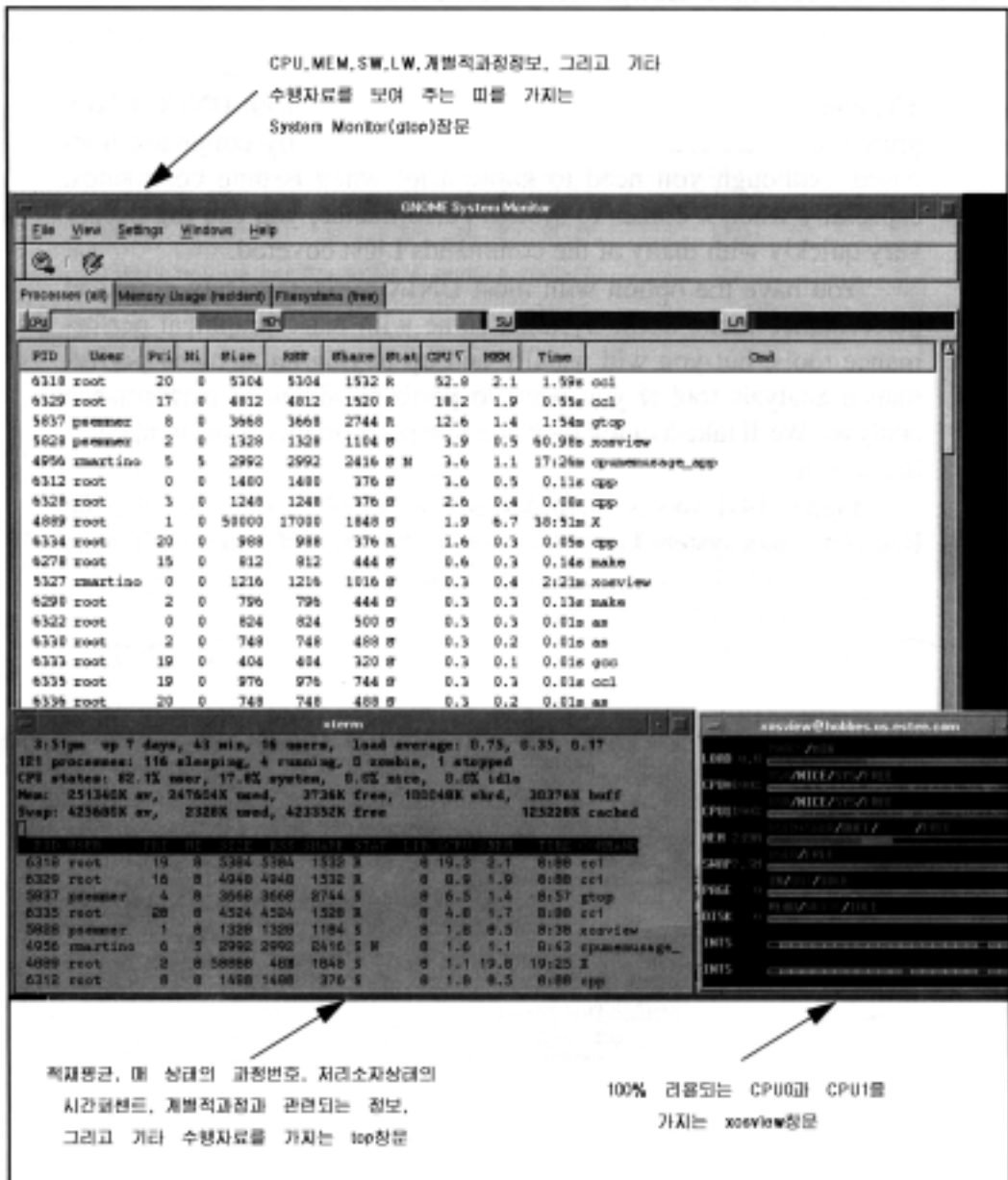



그림 14-1. Red Hat Linux 수행도구화면

표시된 유용한 top 체계의 자료가운데는 다음과 같은것들이 있다.

- 마지막 1, 5, 15 분에서 평균부하
- 존재하는 프로쎄스들의 개수와 매 상태에서 프로쎄스들의 개수
- 체계에서 처리장치당 매 처리장치상태에서 보낸 시간의 퍼센트

이와 같은 통보는 체계모니터창문안에 top 와 xosview 창문으로 점차 포함되여 가고 있다.

다음에 top 창문에서는 사용자의 기억, 자유로운 기억, 공유된 기억을 포함하여 기억자료를 보여 준다.

자료는 다음과 같은것을 포함하는 ps 와 류사한 형식으로 개별적프로쎄스에 대해서 제공된다.

PID	프로쎄스 ID 인 수값
USER	프로쎄스의 소유자의 이름
PRI	프로쎄스의 현재 우선권
NI	-20 부터 +20 까지 범위에서 좋은 값
SIZE	KByte 로 프로쎄스의 총 크기
RSS	KByte 로 프로쎄스의 상주크기
STATE	프로쎄스의 현재상태
TIME	체계번호와 CPU 를 리용한 프로쎄스의 초
%CPU	CPU 퍼센트
%MEM	기억퍼센트
COMMAND	현재 수행하는 프로쎄스의 지령이름

앞에서 고찰한 대부분의 지령들에서와 같이 top 도 UNIX 방안에 따라 차이난다. 다른 UNIX 방안들에서 일부 다른 마당들도 있을수 있다. 여기서 이 장에서 top 를 수행시킨 체계는 보통 될수록 빨리 볼수 있도록 임의의 UNIX 체계들에서도 할수 있는 형식을 취하고 있다. 여기서 수행시간 top 의 대부분방안은 기호기초응용프로그람이고 그것을 수행시키는데 도형말단이 요구되지 않는다. 이 실례는 X 말단에서 기호방식으로 top 를 수행시키고 있다.

이 실례에서 사용자의 체계는 두개의 CPU 를 가진다. 만일 xosview 창문에서 주의하여 본다면 CPU 0 과 CPU 1 은 둘다 100%리용되였다는것을 알수 있을것이다. 이 화면장면을 얻는 시간에 이 체계에서는 Linux 핵심부를 편작하였고 짧은 시간주기동안 체계에서 CPU 의 모든 자원을 리용하였다. 핵심부의 편작에 리용된 ccl 프로그람을 top 와 체계모니터창문 둘다에서 볼수 있고 체계에서 CPU 자원의 실제적자원을 전부 리용하였다.

그림 14-1 은 어떻게 체계자원이 리용되였는가를 보는데 필요한 어떤 다른 도구들이 있는가를 보여 준다. xosview 는 얼마나 많은 체계자원들이 리용되였는가 하는 도형적인 개괄과 빠른 참조를 제공한다. top 와 체계모니터는 대부분의 체계자원을 리용한 프로쎄스를 결정하는데 리용될수 있다.

HP GlancePlus/UX

체계가 무엇을 요구하고 있는가에 대한 더 깊은 리해를 하려면 UNIX 의 지령들을 리용할수 있다. 일정한 시간동안 체계에서 무엇을 하는가에 대한 자료를 알려고 한다면 UNIX 의 어떤 지령을 리용하면 될것이다. 같은 프로쎄스에 대하여 더 알아야 할 자료가 있다 해도 다른 추가적지령을 리용하지 않을수 있다.

이제 서술하는 다른 기술들은 실제시간에 유용한 자료를 얻는 방법을 제시하기 위한 수단들이다. 이것은 지령과 프로쎄스를 조사하게 하지만 통보를 보는데는 지나친 관심을 하지 않게 한다. 이 도구가 바로 HP GlancePlus/UX 이다. 이 도구는 Solaris, HP-UX, AIX 를 포함하여 여러개의 UNIX 방안들에서 수행할수 있다.

GlancePlus 는 기호방식 혹은 그라프방식에서 수행시킬수 있다. 이것은 그라프 혹은 기호기초방식으로 임의의 화면장치에서 수행시킬수 있으므로 GlancePlus 의 기호기초방식을 리용하기로 한다. 왜냐하면 GlancePlus 의 Motif 방식에서 리용되는 많은 색들이 실례들에서 볼수 없는것과도 관련되기때문이다. 여기에서는 기호방식을 리용할 때 실례들에서 더 명백히 표시해 준다. 필요한것을 보려면 GlancePlus 의 두 방안을 리용해 보는것이 좋다.

실례에서 리용된 체계는 8 개 처리장치들, 4GByte 의 RAM, 그것과 접속된 EMC 대칭디스크의 필요한 용량을 보장해 준다.

그림 14−2 는 GlancePlus 와 관련되는 여러개의 화면가운에서 한개의 화면을 보여 주고 있다. 이 화면은 Process List 화면이고 Global 화면도 참고한다. 이 화면은 GlancePlus 를 기동시켰을 때 기정화면이다.

그림 14−2 에서 보여 준 화면의 두 특징은 일정한 가치를 가진다. 즉

① 화면의 꼭대기에서 4 개의 **조직기입**(histogram)은 수값들의 렬들보다 알아보기 더 쉬운 형식으로 CPU, Disk, Memory, Swap Utilizatin 의 도형적표시를 주고 있다.

② Process Summary 는 ps -ef 와 류사한 렬들을 가지는데 많은 체계관리자들의 기능에서 류사한 렬들이다. GlancePlus 는 대단히 적은 자원들을 리용하면서도 출발점만 규정해 주면 필요한 프로쎄스들만 선택할수 있는 능력을 준다.

GlancePlus 를 리용하면 다음과 같은 내용들을 자세히 알아 볼수 있다.

- 프로쎄스목록
- CPU 통보
- 기억기통보
- 교환공간
- 디스크통보

- LAN 상세한 내용
- NFS 체계
- PRM(프로쎄스자원관리자)개요
- 파일체계에 의한 I/O
- 디스크에 의한 I/O
- 론리기록권에 의한 I/O
- 체계표들

그림 14-2 는 GlancePlus 화면의 한 장면이다.

그림 14-2. HP GlancePius/UX 처리목록화면

실례에서 보여 준 프로쎄스목록은 체계자원들이 가장 높은 수준에서 리용되고 있다는것을 보여 준다. GlancePlus 를 표시하는데 휴대형콤퓨터에서 말단모방자를 리용하기로 한다. 많은 체계관리자들은 PC 를 리용하고 UNIX 관리기능을 수행하는데 말단모방자를 리용하려 하고 있다. 이 화면에서 보여 준 정보는 임의의 시간에 선택에 의하여 갱신될수 있다는것을 알아야 한다. 만일 체계가 불변(steady)상태방식에서 수행하고 있다면 많은 변경을 요구할수 없으므로 오랜 시간구간이 요구될수 있다. 그러나 동적인 환경을 가질수 있고 조직기입과 잠간동안 갱신되는 다른 정보를 볼것을 요구할수 있다. 다른 경우

에는 필요할 때 갱신구간을 변경시킬수도 있다. 화면의 밑에 배치된 기능건들을 리용하여 다른 기능의 수행에로 넘어 갈수도 있다.

프로쎄스목록서술

프로쎄스목록화면은 체계자원과 능동프로쎄스들의 상태에 대한 개괄을 보여 준다.

화면의 꼭대기부분(조직기입부분)은 GlancePlus 의 많은 화면들에서 공통적인 부분이다. 화면의 밑부분은 능동프로쎄스들의 개요를 표시한다.

행 1 은 GlancePlus 의 개발과 판본번호, 시간, 체계의 이름, 체계의 형을 제공한다. 여기서는 GlancePlus 의 11.01 방안을 리용하고 있다.

행 3 은 CPU 의 전반상태에 대한 정보를 제공한다. 이것은 조작자가 체계에 대하여 알려고 하는 중요한 대부분의 정보를 포함하고 있다. 즉 자기의 CPU 가 혹사되였는가 하는것도 보고 있다.

CPU 의 보조리용띠는 다음과 같은 부분들로 구분되여 있다.

① S 는 문맥절환과 체계호출과 같은 체계의 능동상태에서 보낸 총 시간을 지적한다.
② N 은 좋은 사용자프로쎄스들의 수행(이것은 낮은 수준에서 수행한다.)으로 보낸 총 시간을 지적한다.
③ U 는 사용자프로쎄스의 수행으로 보낸 총 시간을 지적한다.
④ R 는 실시간프로쎄스들을 지적한다.
⑤ A 는 좋지 않은 우선권을 가지고 프로쎄스들의 수행에 리용된 총 시간을 지적한다.

행 3 의 오른쪽에서는 CPU 리용의 퍼센트를 보여 준다. 만일 체계가 CPU-제한을 받는다면 항상 이 수값이 100%에 가깝다는것을 알수 있을것이다. 여기에서는 Current(현재), Average(해석을 시작한 후에 평균), High(높은)에 대한 통계자료를 볼수 있다.

행 4 는 가장 긴장하게 설치된 디스크의 리용을 보여 준다. 이 조직기입은 갱신기간에 파일체계와 가상기억디스크의 I/O 의 퍼센트를 지적한다. 이 조직기입은 두 부분으로 구분되여 있다.

① F 는 사용자가 읽고 쓸 때 파일체계의 능동성과 다른 비페지화능동성의 총 량을 지적한다.
② V 는 페지화의 가상기억에 봉사된 디스크 I/O 의 퍼센트를 지적한다.

Current, Avg 와 High 통계자료는 CPU 리용해설에서와 같은 의미를 가진다.

행 5 는 체계기억의 리용을 보여 준다. 이 조직기입은 세개 부분으로 나누어 져 있다.

① S 는 리용하는 체계에서 봉사와 기억의 총량을 지적한다.
② U 는 사용자프로그람들과 자료에 봉사된 기억의 총량을 표시한다.
③ B 는 고속완충기에 봉사된 기억의 총량을 지적한다.

Current, Avg 와 High 통계자료는 CPU 리용해설에서와 같은 의미를 가진다.

행 6 은 교환리용정보를 보여 주는데 두 부분으로 구분된다.

① R는 예정되였지만 리용되지 않는다는것을 지적한다.

② U는 리용되는 교환공간을 지적한다.

이 모든 세개의 령역(CPU, Memory, Disk)은 각각 기능건 F2, F3, F4를 리용하여 더 해석할수도 있다. 또한 수행하고 있는 GlancePlus의 방안에 따라서 다른 기능건들도 있을수 있다. 이 건들중 한개 건을 선택하면 Process List 화면으로부터 선택된 령역에서 더 많은 기능을 제공하는 다른 화면으로 이동하게 된다. 이런 더 자세한 화면들은 많은 다른 체계들에서도 리용될수 있다. Process List 화면에서 제시되지 않은 대부분의 내용들은 CPU, Memory, Disk 화면에서 볼수 있다. 여기서는 이 내용들에 대하여 자세히 해설하기로 한다.

Process List 화면의 밑에서는 체계에서 수행하고 있는 능동적프로쎄스들을 보여 준다. UNIX 체계에서 수행하는 대표적인 많은 프로쎄스들이 있으므로 CPU리용의 출발점의 지령 o를 리용해야 한다. 만일 실례로 5%로 출발점을 설정한다면 그 구간에서 5%의 평균 CPU 리용을 초과하는 프로쎄스들만 표시될것이다. 리용된 RAM의 용량(상주크기)과 같은 규정할수 있는 출발점의 다른 형들도 있다. 만일 출발점을 규정한다면 가장 관심 있는 프로쎄스들만 볼수 있다. 즉 가장 많은 체계자원들을 리용하는 프로쎄스들만 볼수도 있다.

정의된 출발점에 대한 요구를 가지는 매 능동프로쎄스의 행도 있다. 이때 표시해야 할 프로쎄스들에 대하여서는 한페지이상씩 차지할수도 있다. 화면의 밑에서 오른쪽구석에 있는 통보는 페지가 계속된다는것을 지적한다. f에 의해서 다음 페지를 보려면 앞으로 정보를 밀수도 있고 b에 의하여 뒤로 한페지를 밀수도 있다. 보통 일부 프로쎄스들만 체계자원들의 대부분을 리용하므로 매 프로쎄스가 한페지만 차지하도록 출발점을 설정하는것이 좋다. GlancePlus에서 리용할수 있는 모든 지령들도 있다. 마지막그림은 GlancePlus에 의하여 인식되는 지령들을 보여 준다.

아래에서는 프로쎄스머리부에 대한 간단한 개괄을 준다.

Process Name	수행할 프로그람을 적재할 때 리용할 이름 혹은 략자.
PID	프로쎄스의 식별번호
PPID	선조프로쎄스의 PID
Pri	프로쎄스의 우선권. 작은 수값일수록 우선권은 더 크다. 체계수준프로쎄스들은 보통 0과 127 사이의 값을 가지고 수행된다. 다른 프로쎄스들은 보통 128과 255 사이의 값을 가지고 수행된다. 좋은 프로쎄스는 가장 낮은 우선권인데 이 우선권은 가장 큰 수값을 가진다.
User Name	프로쎄스를 시작한 사용자이름
Cpu Util	첫째 수값은 갱신된 구간에서 이 프로쎄스가 리용한 CPU의 퍼센트. 8개의 처리장치체계인 경우에는 최대로 800%까지 될수

	있다. 둘째 수값은 GlancePlus 가 기동된 후에 이 프로쎄스가 리용한 CPU 의 퍼센트이다. 많은 조직자들은 작은 갱신구간으로 체계에서 련속 수행하도록 GlancePlus 를 남겨 둔다. GlancePlus 는 대단히 작은 체계비용을 리용하므로 초과비용은 거의 없다.
Cum CPU	프로쎄스가 리용한 총 CPU 시간. GlancePlus 는 정보를 수집하기 위하여 중간출력을 리용한다. 만일 프로쎄스가 시작되기전에 중간출력이 시작되였다면 프로쎄스가 리용한 루적된 정확한 CPU 시간을 얻을것이다.
Disk IO Rate	첫째 수값은 마지막으로 갱신된 구간에서 초당 평균 디스크의 I/O 비률이고 둘째 수값은 GlancePlus 가 시작되였거나 프로쎄스가 시작된 후에 평균디스크의 I/O 비률. 디스크의 I/O 는 많은 다른 의미도 가진다. 디스크의 I/O 는 처음으로 디스크에서 자료가 없는 블로크를 찾고 RAM 으로부터 자료를 써넣거나 완전히 페지화 및 교환을 해야 한다는것을 의미한다. 일부 프로쎄스들은 다른 연산보다 더 많은 디스크의 I/O 를 요구할수도 있다. 이 수가 대단히 클 때 충분한 RAM 을 가지는가, 못가지는가를 면밀히 조사해야 한다. vhand 프로쎄스는 피동 혹은 교환과 같은 페지출력능동성을 항상 가진다.
RSS Size	프로쎄스가 소비한 키로바이트(Kbyte)단위의 RAM 의 용량. 이 용량을 **상주크기**(Resident Size)라고 부른다. RAM 에 있는 프로쎄스와 관련된 모든것은 이 RSS Size 에 포함되는데 여기에는 프로쎄스의 자료, 탄창, 본문과 공유된 기억토막들과 같은것들이 있는데 검사하기에 편리한것들이다. CPU-제한된 체계는 느리게 작업한다는것이 분명하므로 항상 기본응용프로그람이 리용하는 RAM 용량의 표시를 무시해도 된다. 일부 응용프로그람들이 적은 RAM 용량을 리용하면서 큰 자료모임을 처리할 때에는 RAM 이 과부하를 받게 된다. 이 렬은 현재 프로쎄스가 리용하고 있는 RAM 을 보여 준다.
Block on	프로쎄스가 봉쇄된 리유(수행할수 없는 리유). 만일 프로쎄스가 현재 봉쇄되여 있다면 왜 그런가를 알려 준다. 만일 프로쎄스가 수행되고 있다면 마지막에 봉쇄되였던 원인을 보여 준다. 아래의 Thd Cnt 는 봉쇄된 프로쎄스의 가장 보편적인 원인들의 목록이다.
Thd Cnt	현재 프로쎄스의 수행의 길(thread)들의 총 개수
략자	**봉쇄된 프로쎄스의 리유**
CACHE	리용할수 있을 때까지 고속기억완충기의 대기
DISK	디스크연산이 완성될 때까지 대기

INODE	내부연산을 완성할 때까지 대기
IO	디스크가 아닌 I/O를 완성할 때까지 대기
IPC	공유기억연산을 완성할 때까지 대기
LAN	LAN 연산이 완성될 때까지 대기
MESG	통보순서연산이 완성될 때까지 대기
NFS	NFS 요청이 완성될 때가지 대기
PIPE	파이프에로 그리고 파이프로부터 자료의 대기
PRI	보다 높은 우선권을 가진 프로쎄스의 수행에 의한 대기
RFA	원격파일접근이 완성될 때까지 대기
SEM	기발을 리용할수 있게 될 때까지 대기
SLEEP	프로쎄스가 sleep 혹은 wait 호출에 의한 대기
SOCKT	소케트연산이 완성될 때까지 대기
SYS	체계자원에 대한 대기
TERM	말단전송의 대기
VM	가상기억연산을 완성할 때까지 대기
OTHER	GlancePlus 가 결정할수 없는 리유로 대기

CPU 통보화면서술

만일 Process List 화면이 CPU 가 과부하상태였다는것을 지적한다면 그림 14-3 에서 보여 준 CPU Report 화면을 보아야 할것이다. 이 화면은 GlancePlus 통보의 7 개의 상태의 형에 대한 유용한 정보를 제공하고 있다.

5 개의 상태의 형가운데서 매개는 추가적정보를 제공하는 렬들이다.

Current	마지막시간구간에 이 상태로 봉사된 CPU 시간의 퍼센트를 표시한다.
Average	GlancePlus 가 기동된 후에 이 상태에서 보낸 CPU 시간의 평균 퍼센트를 표시한다.
High	GlancePlus 가 기동된 후에 이 상태에서 봉사된 CPU 시간의 가장 큰 퍼센트를 표시한다.
Time	마지막구간에서 이 상태에서 지나간 CPU 시간을 표시한다.
Cum Time	GlancePlus 가 기동된 후에 이 상태에서 지나간 CPU 시간의 총 합을 표시한다.

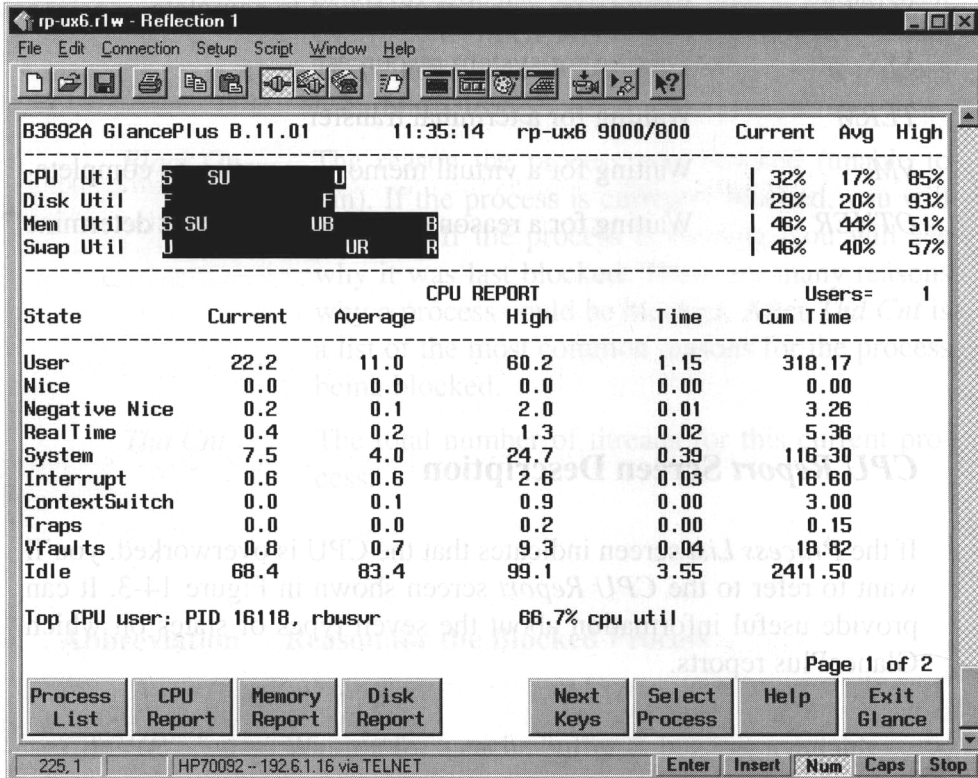



그림 14-3. HP GlancePlus/UX CPU Report 화면

10개 상태에 대한 해설을 주면 다음과 같다.

User	표준우선권에서 수행하는 사용자능동상태에서 보낸 CPU시간
Nice	좋은 방식에서 사용자코드를 수행하는데 보낸 CPU시간
Negative Nice	높은 우선권에서 코드를 수행하는데 보낸 CPU시간
Realtime	높은 우선권에서 수행하는 실시간프로세스들을 수행하는데 보낸 CPU시간
System	체계호출과 프로그람들의 수행에 리용된 CPU시간
Interrupt	체계중단수행에 소비한 CPU 시간. 여기서 큰 값은 페지화 및 교환과 같은 많은 I/O의 수행을 지적할수 있다.
ContSwitch	프로쎄스들사이에 문맥절환에 리용한 CPU시간
Traps	함정을 조종하는데 리용한 CPU시간
Vfaults	페지실패조종에 리용한 CPU시간
Idle	아무것도 하지 않으면서 보낸 CPU시간

CPU Report 화면은 체계의 수행순서의 길이 혹은 평균부하도 보여 준다. 이것은

CPU Report 화면의 둘째 페지에 표시된다. CPU 는 어떤 사건의 대기로 수행가능한 프로쎄스들의 개수에 대한 Current, Average, High 값들을 보여 준다. 체계가 놀고 있는 상태일 때에는 체계수행의 순서길이의 표준값을 알수 있고 체계가 표준적으로 리용되고 있을 때에는 본 값과 이 표준값을 비교할수 있다.

CPU Report 화면에서 통보된 마지막령역은 평균부하, 체계호출, 중단, 문맥절환들이다. 이 값이 크다면 문제는 표준상태이고 문제를 제기할수 없기때문에 이것을 자세히 볼 필요는 없다. 만일 문제를 수정한다면 이 값들이 감소한다는것을 볼수 있을것이다.

그림 14-4 에서 보여 준것처럼 체계에서 전체 CPU 들의 상태를 보기 위하여 GlancePlus 를 리용할수 있다. 이것은 8 개의 처리장치체계를 보여 준것이다.

그림 14-4. GlancePlus 에서 전체 CPU 들의 상태

기억기통보화면서술

그림 14-5 에서 보여 준 Memory Report 화면은 기억관리사건의 여러가지 형태에 대한 정보를 제공한다. 보여 준 통계자료들은 퍼센트가 아니라 계수기값 형식이다. 대부분 놀고 있는 체계에 대한 이 계수기값을 보면 다음에 체계의 부하가 무엇때문에 급속히 변동되는가를 고찰할수 있다. 이 실례에서 보여 준 자료들은 CPU 보다 기억이 **병목**(Bottleneck)으로 되고 있다는것을 보여 준다.

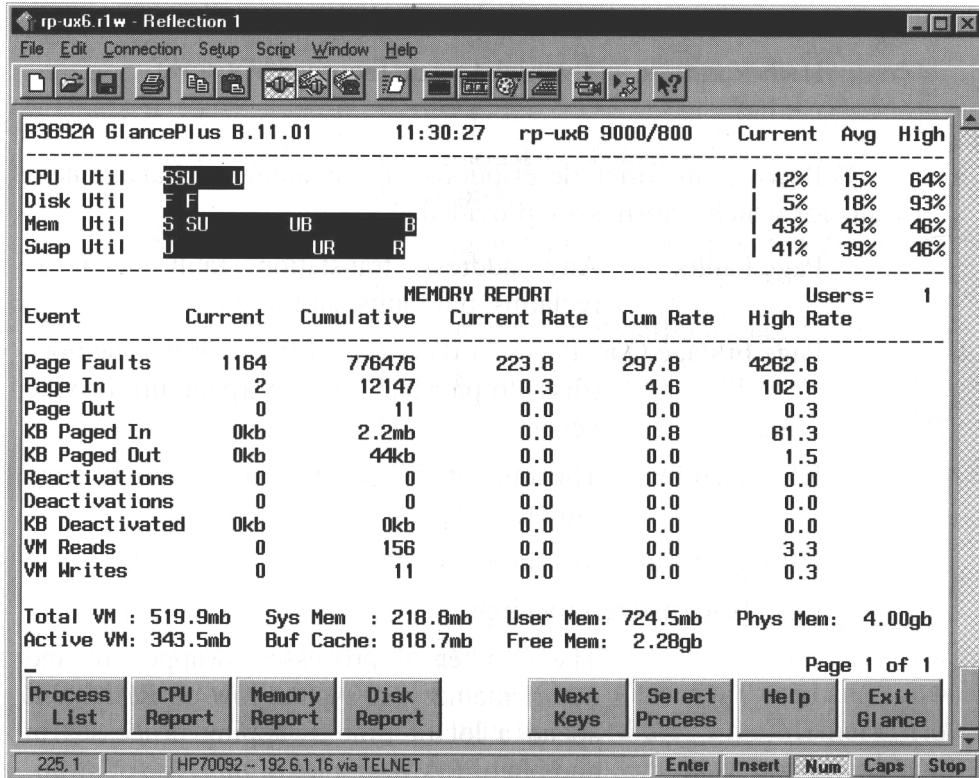



그림 14-5. HP GlancePlus/UX Memory Report 화면

다음의 5개 통계자료들은 매개 기억관리사건에 대하여 보여 준다.

Current	마지막구간에서 나타난 한가지 사건의 회수. 만일 구간을 갱신한다면 계수기값이 변하므로 적당한 시간구간으로 고착시킬수 있다.
Cumulative	GlancePlus가 기동된후에 이 사건들의 모든 계수기의 합
Current Rate	초당 사건개수
Cum Rate	수집된 구간모임에서 평균비률
High Rate	기록된 가장 큰 비률

아래에서는 통계자료가 제공된 기억관리사건에 대한 간단한 해설을 준다.

Page Fault	교정, 사소한 실패와 같은 임의의 주소변환실패
Page In/Page Out	가상기억(디스크)으로부터 물리적기억(page in)에로 혹은 거꾸로 이동되는 자료의 페지들
KB Paged In	페지실패로 페지화된 자료의 총합
KB Paged Out	디스크에서 나간 페지화된 자료의 총합
Reactivations/Deactivations	기억에로 또는 기억으로부터 교환된 프로쎄스들의 개수. RAM 에

	서 체계의 아래 부분은 RAM 안으로 또는 밖으로 프로쎄스들의 교환에 많은 시간을 소비할것이다. 만일 이 형태의 교환이 많이 진행된다면 CPU 의 리용률은 높아 질것이고 다른 일부 통계자료들도 증가된다는것을 알수 있을것이다. 이것은 교환이 많이 진행된다는 징표로 된다.
KB Reactivated	불충분한 RAM 으로 하여 한 프로쎄스는 밖으로 다른 한 프로쎄스는 RAM 의 안으로 교환된 정보의 총
KB Deactivated	프로쎄스들이 디스크에로 이동될 때 밖으로 교환되는 정보의 총합
VM Reads	디스크에 읽어 넣은 가상기억의 총합. 이 수가 크면 클수록 체계는 더 자주 디스크에서 처리를 한다.
VM Writes	기억관리 I/O 의 총합

다음의 값들은 기억화면에서도 리용된다.

Total VM	모든 프로쎄스들에서 리용된 가상기억의 총합
Active VM	모든 능동프로쎄스들에서 리용된 가상기억의 총합
Sys Mem	리용하는 체계에 봉사된 기억의 총합
Buf Cache Size	고속완충기의 현재 크기
User Mem	사용자에게 봉사되는 기억의 총합
Free Memory	현재 리용되지 않은 RAM 용량
Phys Memory	체계에서 RAM 의 총량

이 화면은 리용되는 기억의 부분체계가 어떠한가에 대한 많은 정보를 준다. 체계가 놀고 있는 상태에 있을 때와 큰 부하를 받고 있을 때 통계자료를 보고 두 자료를 비교할수 있다. 기록에서 일부 좋은 수값들은 Free Memory(어떤 조건에서 어떤 리유로 RAM 을 배치받지 않았는가를 알려면)와 Total VM(모든 프로쎄스들에 배치된 가상기억이 얼마나 큰가를 보기 위해서)에 있다. RAM 용량이 큰 체계는 리용할수 있는 기억을 많이 가질것이고 RAM 용량이 작은 체계는 많은 가상기억을 가질것이다.

디스크통보화면서술

Disk Report 화면은 그림 14-6 에서 보여 주고 있다. 여기서 local 과 remote 정보의 그룹들을 볼수 있다.

그림에서는 국부체계에 설치된 모든 디스크들에로 론리적 및 물리적접근에 관한 각각 8 개 사건별로 디스크통계자료들을 주고 있다. 이 사건들은 체계에서 취급한 모든 디스크의 작업상태를 반영하고 있다.

여기서는 제공된 8 개 디스크통계자료를 해설한다.

Requests	마지막구간에서 그 형을 요청하게 될 총수
%	다른 형들에 관한 디스크에서 이 형의 사건의 퍼센트

Rate 초당 이 형의 요청된 평균수
Bytes 마지막구간에서 이 사건으로 전송된 총 바이트들의 수
Cum Req GlancePlus가 전송된 후에 요청을 축적한 회수
% GlancePlus 가 기동된 후에 이 형태의 디스크사건의 상대적퍼센트
Cum Rate 축적된 구간에서 평균비률
Cum Bytes GlancePlus 가 기동된 후에 이 형태의 사건이 전송된 총 바이트들의 수

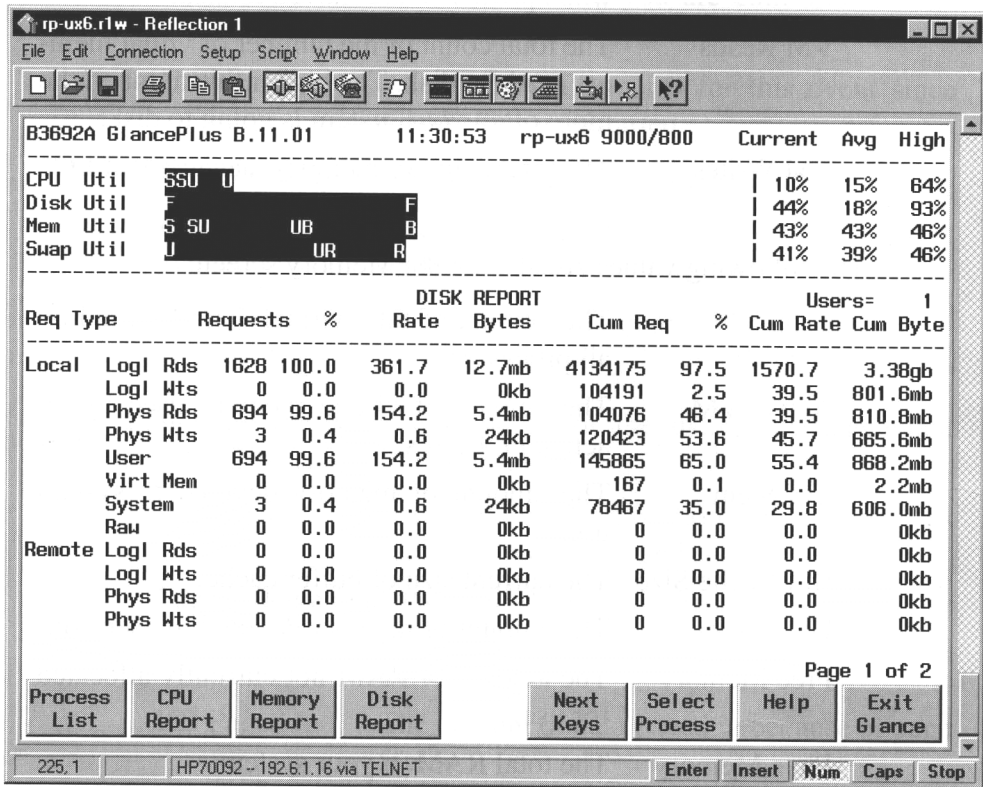



그림 14-6. HP GlancePlus/UX Disk Report 화면

다음으로 체계에서 Local 밑에 목록화된 통계자료에 대한 디스크사건들에 대하여 해설한다.

Logl Rds and Logl Wts 디스크에로 론리적읽기 및 쓰기의 개수. 디스크는 표준적인 고속완충기로 리용되므로 론리적읽기는 디스크에로의 물리적접근을 요구하지 않을수 있다.

Rhys Rds 디스크에서 물리적읽기의 개수. 이 물리적읽기는 파일체계의 론리적읽기 혹은 가상기억관리에 관계될수 있다.

Phys Wts	디스크에 물리적쓰기의 개수. 이것은 파일체계의 능동성 혹은 가상기억관리에 기인될수 있다.
User	사용자의 파일 I/O 조작의 결과를 물리적디스크 I/O 의 총계
Virtual Mem	가상기억관리능동성에 의한 물리적디스크 I/O 의 총계
System	내부조작갱신과 같은 체계자체의 I/O
Raw	미숙한 방식의 디스크 I/O 의 총계

많은 디스크능동성은 NFS 의 설치된 디스크에 의하여 진행될수도 있다. 통계자료는 Remote 디스크에 대해서도 볼수 있다.

디스크접근은 모든 체계들에서 요구된다. 다음과 같은 문제점이 있을수 있다. 디스크능동화는 무엇때문에 불필요하고 체계를 더디게 하는가? 시작할 때 좋은 방법은 "User"디스크 I/O 의 총 수를 Virtual Mem 디스크 I/O 의 총 수와 비교하여 보는것이다. 만일 체계가 사용자 I/O 보다 가상기억 I/O 를 더 많이 수행하고 있다면 필요한만큼 기억리용상태를 조사해 보아야 할것이다.

GlancePlus 요약

이미 서술된 프로쎄스목록, 전역적통계자료, 화면과 CPU, 기억과 디스크화면들외에 다음과 같은 많은 화면들도 있다.

Swap Space	모든 교환령역의 상세한 정보를 보여 준다. 다른 **발행**(Release)에서는 다른 이름으로도 부르고 있다.
Network By Interface	체계에서 모형화된 매 LAN 카드에 대한 상세한 정보를 준다. 이 화면은 다른 발행에서는 다른 이름을 가질수도 있다.
NFS Global	귀환 및 외부로 나가는 NFS 가 설치된 파일체계에 대한 상세한 정보를 제공한다. 다른 발행에서는 다른 이름으로 불리워 질수도 있다.
Select Process	조사한 단일한 프로쎄스를 선택하게 한다. 다른 발행에서는 다른 이름을 가질수 있다.
I/O By File Sys	설치된 매 디스크분구에서 I/O 의 상태를 보여 준다.
I/O By Disk	설치된 매 디스크에서 I/O 의 상세한 정보를 보여 준다.
I/O By Log1 Vol	설치된 매개의 론리기록권에서 I/O 의 상세한 정보를 보여 준다.
System Tables	내부적체계표의 상세한 정보를 보여 준다.
Process Threshold	프로쎄스목록화면에서 어떤 프로쎄스들이 표시될것인가를 정의한다. Global 화면과 같이 다른 발행에서는 다른 이름으로 될수 있다.

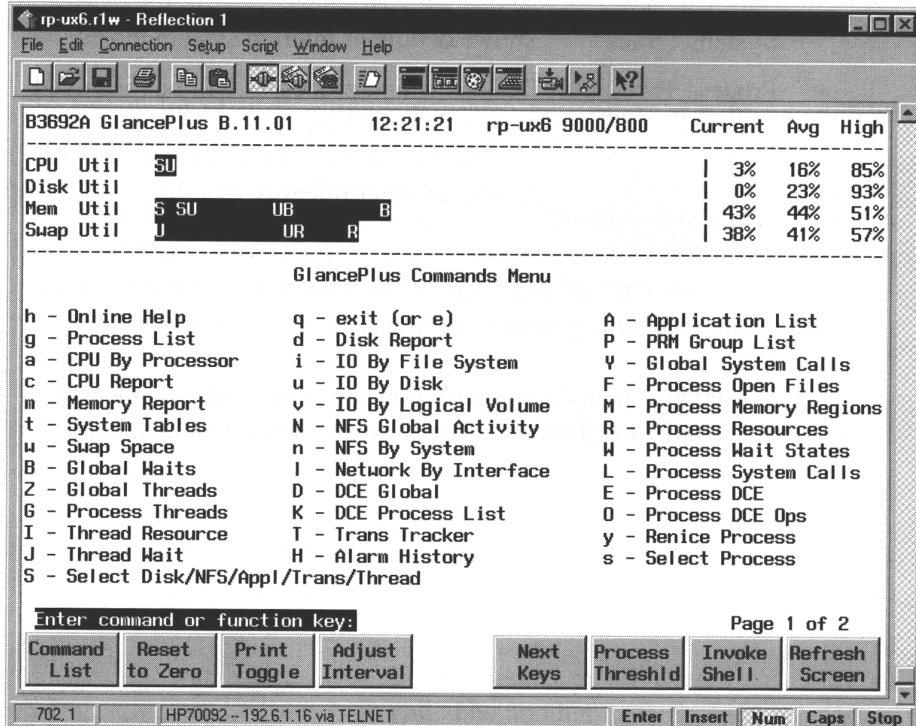



그림 14-7. HP GlancePlus/UX Command List 화면 1

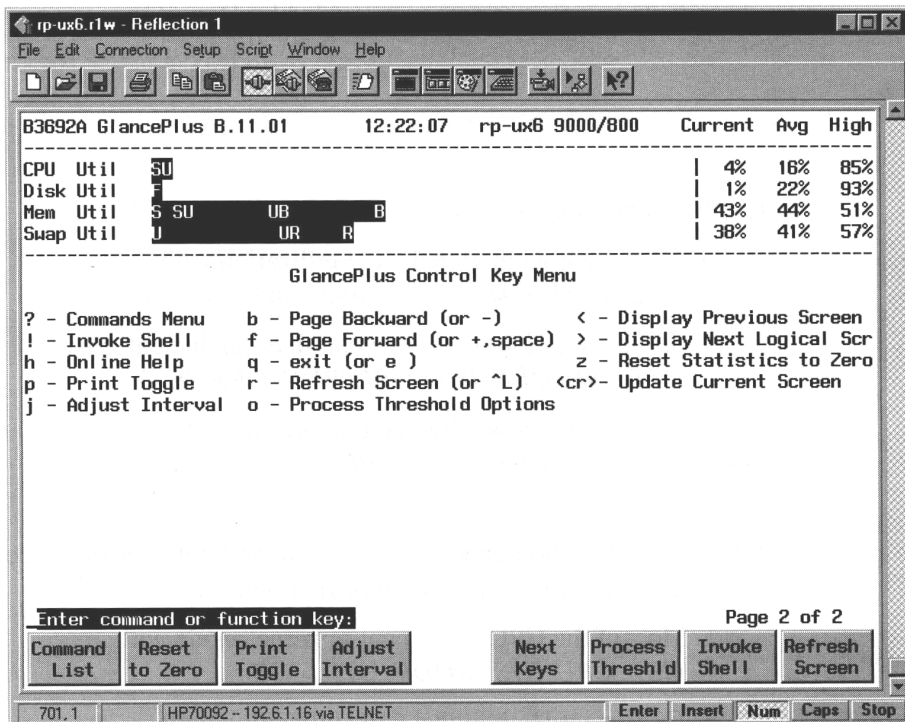



그림 14-8. HP GlancePlus/UX Command List 화면 2

이미 본것처럼 비록 대부분의 공통적으로 리용된 4 개 화면에 대하여 상세히 설명하였지만 앞으로의 체계검사에서는 많은 다른 점들도 있을수 있다.

GlancePlus 안에서는 리용할수 있는 많은 지령들도 있다. 그림 14-7 과 그림 14-8 에서는 GlancePlus 에서의 Command List 화면들을 보여 주고 있다.

병목을 식별하기 위한 VantagePoint 수행대리체의 리용

VantagePoint 수행대리체는 체계에서 병목의 원천을 식별하는데 필요한 체계수행에 관계되는 많은 측정자료들을 볼수 있게 한다. 추적해야 할 측정자료를 얻으려면 gpm 이라는 GlancePlus 의 도형방안을 리용할수 있다. 다음에 gpm 대면부에서 이 도형방안을 리용하여 다른 방법을 리용할수도 있다.

다음으로 체계에서 만날수 있는 병목의 대부분 형태들과 병목의 매 형태와 관계된 측정자료를 고찰하게 된다.

이 정보는 두가지 방법의 수행으로 검증한 Doug Grumann 과 Hewlett Packard 의 Stephen Ciullo 에 의하여 제공된 자료이다.

① VantagePoint 수행대리체의 리용에 의한 CPU 병목

- GBL_CPU_TOTAL_UTIL>90%와 다음공개정보로 모순 없는 높은 전역적 CPU 의 리용한다.
- 중요한 수행순서나 평균부하는 GBL_PRI_QUEUE 혹은 GBL_RUN _QUEUE>3 으로 지적된다.
- PROC_STOP_REASON=PRI 의 우선권으로 봉쇄된 프로쎄스에 대한 정보를 본다.

② VantagePoint 수행대리체리용에 의한 체계 CPU 의 병목(첫 공개정보의 추가는 1 과 같다.)

- CPU 시간의 대부분은 GBL_CPU_SYS_MODE_UTIL>50%로 핵심부방식에서 보낸다.
- GBL_CPU_TOTAL_UTIL>90%와 다음 공개정보로 모순 없는 높은 전역적 CPU 를 리용한다.
- 중요한 수행순서나 평균부하는 GBL_PRI_QUEUE 혹은 GBL_RUN_ QUEUE>3 으로 지적된다.
- PROC_STOP_REASON=PRI 의 우선권으로 봉쇄된 프로쎄스들에 대하여 본다.

③ VantagePoint 수행대리체리용에 의한 문맥절환의 병목 (첫 공개정보의 추가는 2 와 같다.)

- 중요한 CPU 시간은 GBL_CPU_CSWITCH>30%로 절환하는데 보낸다.
- CPU 의 대부분 시간은 GBL_CPU_SYS_MODE_UTIL>50%로 핵심부방식에서

보낸다.

- GBL_CPU_TOTAL_UTIL>90%와 다음 공개정보로 모순 없는 높은 전역적 CPU 를 리용한다.
- 중요한 수행순서 혹은 평균부하는 GBL_PRI_QUEUE 혹은 GBL_RUN_QUEUE>3 으로 지적된다.
- PROC_STOP_REASON=PRI 의 우선권으로 봉쇄된 프로세스들을 본다.

④ VantagePoint 수행대리체리용에 의한 사용자 CPU 병목

- CPU 시간의 대부분은 GBL_CPU_USER_MODE_UTIL>50%로 사용자방식에서 보낸다.
- GBL_CPU_TOTAL_UTIL>90%과 다음 공개정보로 모순 없는 높은 전역적 CPU 를 리용한다.
- 중요한 수행순서 혹은 평균부하는 GBL_PRI_QUEUE 혹은 GBL_RUN_QUEUE>3 으로 지적된다.
- PROC_STOP_REASON=PRI 의 우선권으로 봉쇄된 프로세스들을 본다.

⑤ VantagePoint 수행대리체리용으로 디스크병목

- BYDSK_UTIL>50%으로 적어도 한 디스크장치에서 무모순적으로 리용률이 높다.
- BYDSK_QUEUE>0 으로 령보다 더 큰 길이를 보여 준다.
- PROC_STOP_REASON=CACHE, DISK 혹은 IO 에 의한 리유들로 I/O 로 봉쇄된 프로세스 혹은 수행길을 보여 준다.
- PROC_STOP_REASON=PRI 의 우선권으로 봉쇄된 프로세스들을 보여 준다.

⑥ VantagePoint 수행대리체리용으로 고속완충기병목

- BYDSK_UTIL>25%로 적어도 한 디스크를 적당히 리용한다.
- BYDSK_QUEUE>0 으로 령보다 큰 순서의 길이를 보여 준다.
- GBL_MEM_CACHE_HIT_PCI<90%로 낮은 고속완충기에서 읽은 비트의 퍼센트를 보여 준다.
- PROC_STOP_REASON=CACHE 로 고속완충기로 봉쇄된 프로세스 혹은 수행길들을 보여 준다.

⑦ VantagePoint 수행대리체리용에 의한 기억병목

- GBL_MEM_UTIL>95%로 높은 물리적기억의 리용을 보여 준다.
- GBL_MEM_PAGEOUT_RATE>1 혹은 GBL_MEM_SWAPRATE>0 으로 중요한 페지출력 혹은 임의의 비능동성을 보여 준다.
- Vband 의 PROC_CPU_TOTAL_UTIL>5%로서 vband 프로세스의 무모순적인 능동성을 보여 준다.
- PROC_STOP_REASON=VM 에 의한 가상기억으로 봉쇄된 프로세스 혹은 수

행의 길들을 보여 준다.

⑧ VantagePoint 수행대리체리용에 의한 망병목

- GBL_NET_PACKET_RATE>2 의 평균으로 높은 망파케트의 비률을 본다. 모형에 따라 크게 변할수 있다.
- GBL_NET_OUTQUEUE>0 으로 임의의 출력의 순서화의 진행을 보여 준다.
- PROC _ STOP _ REASON = NFS, LAN, RPC 혹은 SOCKET CBL_NETWORK_SUBSYSTEM_QUEUE> 평균으로 망에서 봉쇄된 프로쎄스 혹은 수행길의 표준개수보다 더 큰가를 보여 준다.
- BYCPU_CPU_INTERRUPT_TIME>30 으로 다른 CPU 들이 거의 놀고 있는 상태에 있을동안 높은 체계방식의 CPU 리용을 가지는 한개의 CPU 를 보여 준다.
- 지뢰리용으로 외부로 나가는 축출 혹은 과도한 충돌의 빈번한 증가를 검사한다.

체계에서 문제를 식별하기 위하여서는 먼저 정상적이고 아무런 문제가 없이 수행될 때의 체계의 자료들을 얻어야 한다. 다음으로 일부 규정 혹은 다른 규정으로 체계가 비정상적으로 수행될 때의 자료와 앞의 자료와 비교하여야 한다.

HP VantagePoint 수행대리체와 HP VantagePoint 수행해석기 /UX

오랜 시간주기마다 추적하는 자료를 표현하는 수행도구들도 있다. 체계관리자들은 흔히 이 련습과제를 **능력계획화**(Capacity Planning)라고 부르고 있다. 능력계획화의 목표는 긴 시간주기동안 어떤 체계자원들이 리용되며 어떤 조절 혹은 추가로 체계에서 앞으로 체계수행의 개선과 계획의 개선을 할수 있겠는가를 결정하는것이다. 여기서는 (형식적으로 Measure Ware Ageut 인)HPVantagePoint 수행대리체와 (PerfView Analyzer 로 리용될)HP VantagePoint 수행해석기 /UX 를 리용할것이고 이와 함께 체계의 수행자료도 볼수 있을것이다. 이 도구들은 HP-UX 에서 수행되며 다른 UNIX 방안에서 수행하는 많은 개선된 도구들과 류사하다.

VantagePoint 수행대리체는 분산된 환경에서 개별적체계에 의하여 설정된다. 개별적체계에서는 자원과 수행의 측정자료를 수집한다. 체계조직에서는 대표적으로 설정되는 VantagePoint 수행해석기 /UX 관리조작대에서는 VantagePoint 수행대리체의 자료를 력사적으로 표시하는데 리용된다. VantagePoint 수행대리체를 리용하여 례외조건들에 제동을 주는 경보를 설정할수도 있다. 실례로 만일 VantagePoint 수행대리체가 CPU 리용이 90%보다 더 큰것과 같은 례외조건을 발견하면 경보통보를 산생시킨다. 경보통보는 다음에 VantagePoint 수행해석기 /UX 에 의하여 표시된다. 앞으로 웃 실례에서의 VantagePoint 수행해석기 /UX 를 리용하기로 한다. 그러나 실지로 이 해석자는 세개의 VantagePoint 수행성분들을 가지고 있다.

Monitor VantagePoint 수행으로부터 경보접수의 능력을 감시하며 필요할 때 경보를 제공하고 표시한다.

Planner	예측을 위한 VantagePoint 수행자료의 보간법으로 능력예측을 제공한다.
Analyzer	다중체계로부터 VantagePoint 수행자료를 해석하고 자료를 표시한다. 동시에 다중체계로부터 자료를 볼수도 있다.

아래의 실례에서는 단일한 체계에서의 작업자료를 보여 준다. 대략 한주동안에 수집된 VantagePoint 수행자료를 취하여 그 일부를 표시해 준다. 이 실례에서는 여러개의 분산된 체계로부터 자료를 택하고 한개의 봉사기만 리용할것이다.

HP VantagePoint 수행대리체는 체계자원리용에 대한 정보를 포함하는 림시파일을 만든다. 오래동안 HPVantagePoint 수행대리체가 수행될수록 작업일지파일에는 더 많은 자료가 기록된다. 이것은 체계를 개선하기 위하여 수행상태가 나쁜 상태를 제거하려고 흔히 호출하는것이다. 일반적으로는 유용한 통보를 얻을 때 충분히 긴 시간주기에서 림시파일을 얻도록 하기 위하여 최소로 한 주동안 HPVantagePoint 수행대리체를 수행시킨다. 일부 체계들에서는 이 시간주기를 월로 취한다. 보통 다른 체계들에서의 한 주는 충분한 시간구간으로 될것이다.

한 주동안에 VantagePoint 수행을 적용하고 그 주에 얻은 체계자원리용의 정도를 보려면 VantagePoint 수행해석기 /UX 를 기동시키면 된다. 그라프는 다시 볼수 있도록 CPU, 기억, 디스크에 대한 자료를 보여 준다. 그림 14-9 는 주간에 전역적 CPU 의 리용상태에 대하여 개괄하여 준다.

VantagePoint 수행해석기 /UX 에 의하여 이 그라프의 매 영상특징을 조절할수 있다. 유감스럽게도 이 그라프에서는 색을 표시하지 않았다. 리용된 색은 콤퓨터화면에서 그라프를 볼 때 파라메터들을 식별하게 한다. 총체적인 CPU 의 리용은 그라프에서 항상 꼭대기점으로 표시되였는데 이것은 체계와 사용자방식의 합을 표시하고 있다.

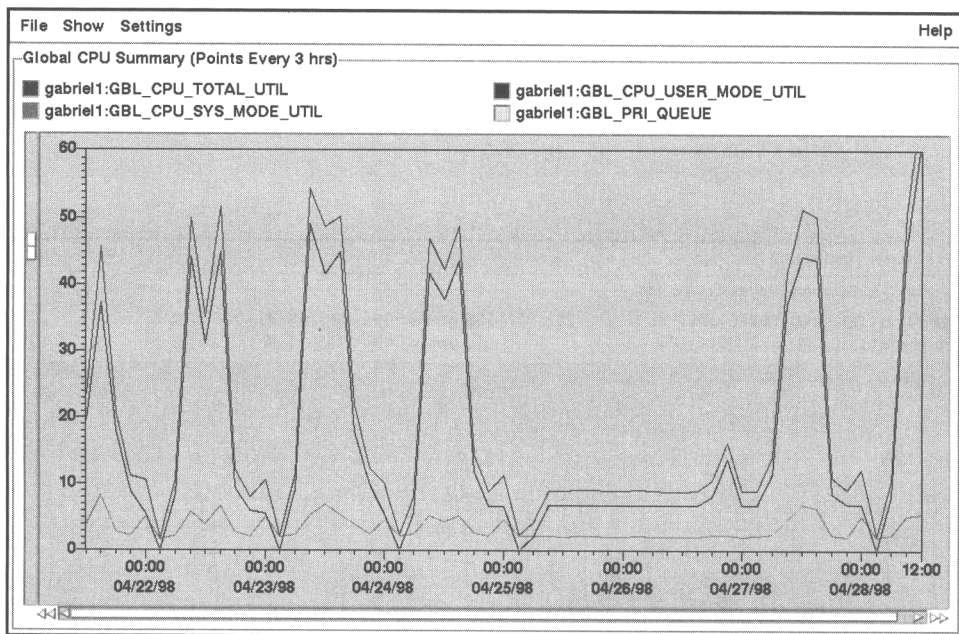



그림 14-9. 전역적 CPU 의 개요화면

그림 14-9는 높은 CPU리용을 반영하는 최초의 시간구간과 낮은 CPU리용을 반영하는 그 나머지 시간구간에서 CPU의 리용을 대비적으로 보여 주고 있다. 어떤 관점에서는 이 그라프를 잘못 리해할수도 있다. 자료점은 매 시간마다 나타나는데 24시간을 주기로 8개의 부분구간으로 고찰된다. 시간이 훨씬 더 작게 취하면 창문으로는 실제적 CPU리용을 볼수 없게 될것이다. 실례에서는 아침의 어떤 시각에 CPU가 긴장하여 리용되는것을 정확히 볼수 있다. 실례에서는 둘째 순간과 셋째 순간사이에서 그것을 볼수 있는데 그 구간은 오전 6시와 9시사이의 긴 시간구간이라는것을 보여 주고 있다. 립자모양과 같이 표시된 다섯째 및 일곱째 순간사이에 명백히 CPU리용이 떨어 진다는것을 알수 있으나 이것은 잘 보지 못한것이다.

그림 14-10은 더 짧은 시간동안의 CPU리용을 창문을 통해서 보여 준다.

이 그림은 시간창문을 통하여 CPU의 리용을 더 명백히 립자모양으로 보여 주고 있다. 창문의 더 좋은 립자모양은 하루동안에 명백히 제시되여 있는데 능동성은 뾰족점으로 나타나 있고 가입점은 오전 8시 30분으로 나타나 있다.

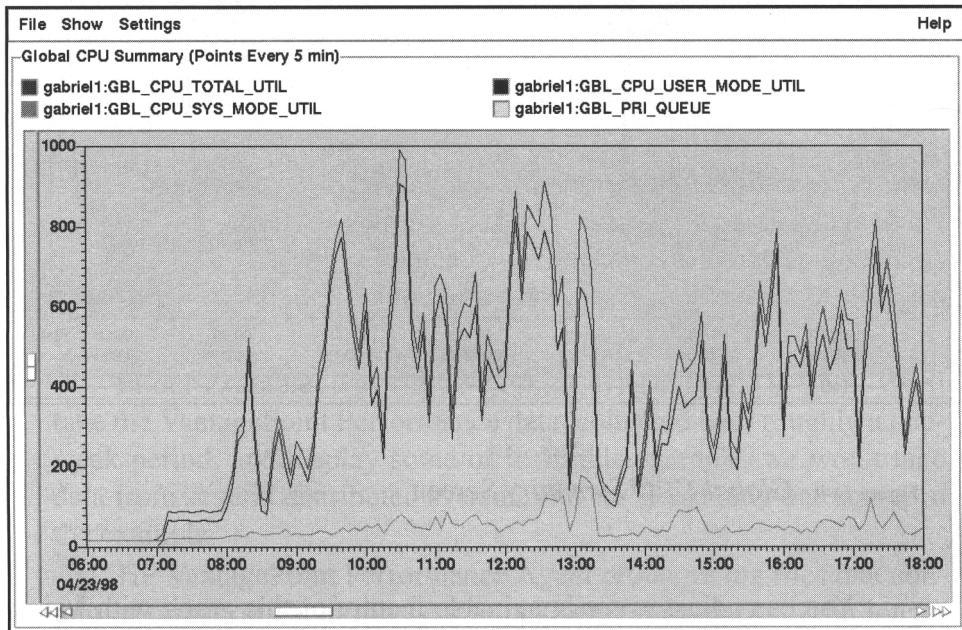



그림 14-10. 전역적 CPU의 개요-짧은 시간주기

기억리용은 그림 14-11에서 보여 준것처럼 주간에 대한 그라프로 표시된다.

사용자의 기억리용은 밑에 있는 그라프로 표시해 주고 있다. 그것은 이미 본 CPU리용에 엄격히 대응한다. 사용자의 기억리용은 비최초시간에 낮고 최초시간에 높다.

체계의 기억리용은 중간에 있는 그라프로 표시해 주고 있는데 주간에는 상당히 안정되여 있다.

총 기억리용은 항상 그라프의 꼭대기선인데 체계 및 사용자기억의 합으로 표시된다. 체계기억리용은 엄격히 갈기때문에 사용자기억은 증가하기도 하고 감소하기도 한다.

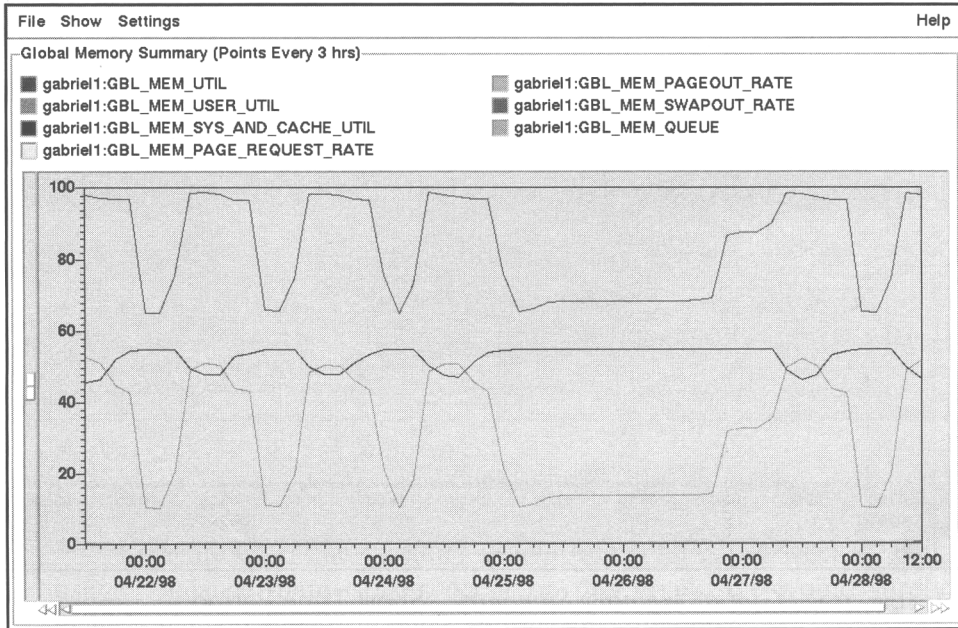



그림 14-11. 전역적기억개요

이 그라프에서 자료점들사이의 3 - 4 시간구간은 요구한 립자모양을 주지 않고 있다. 그림 14-12 는 더 짧은 시간창문에 의한 기억리용을 보여 준다.

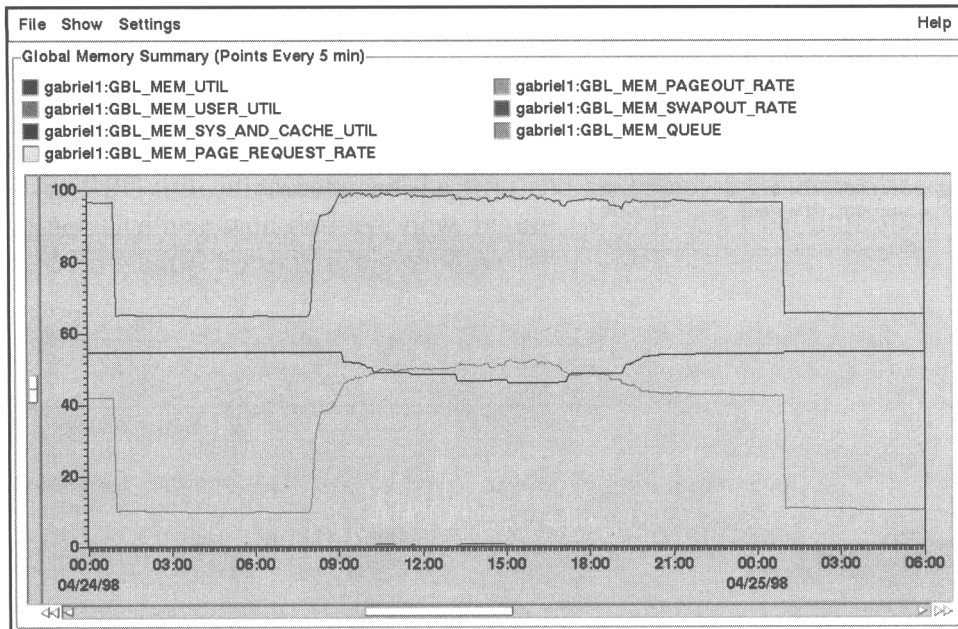



그림 14-12. 전역적기억개요-짧은 시간주기

그림 14-12 는 더 짧은 시간창문에서 기억리용의 명백한 립자모양을 보여 준다. 이

제 우리는 어떻게 기억리용이 엄격히 하루동안에 변하는가를 정확히 볼수 있다.

디스크리용은 그림 14-13에서 보여 준것처럼 주간에 그라프로 표시될수 있다.

이 그라프는 CPU와 기억그라프로 옹근 한주동안의 디스크리용정보를 보여 주고 있다. 많은 뾰족점이 이 그라프에 나타나기때문에 더 짧은 시간창문을 통하여 작은 시간구간들에서 해석해야 한다.

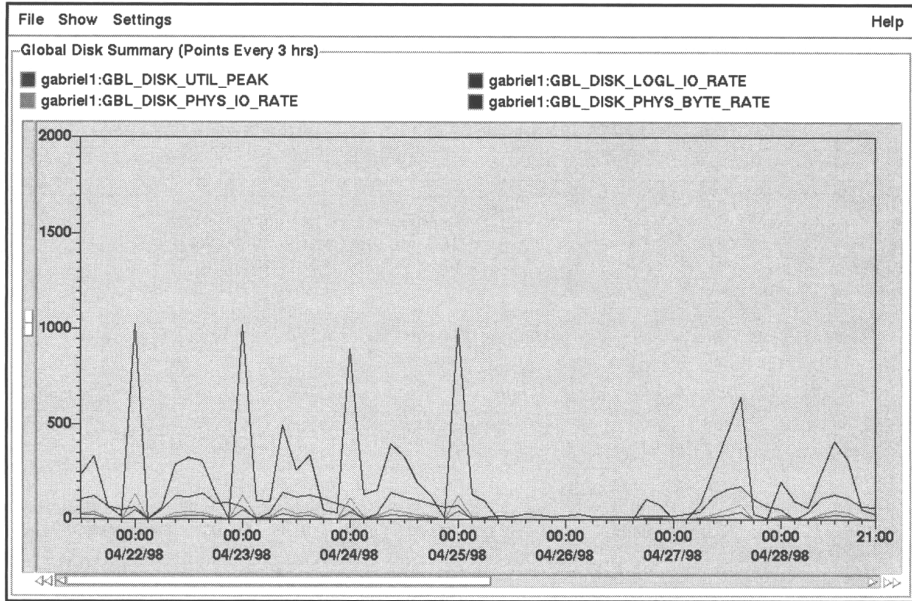



그림 14-13. 전역적디스크개요

그림 14-14는 여러개의 짧은 시간창문을 통하여 디스크리용정보를 보여 준다.

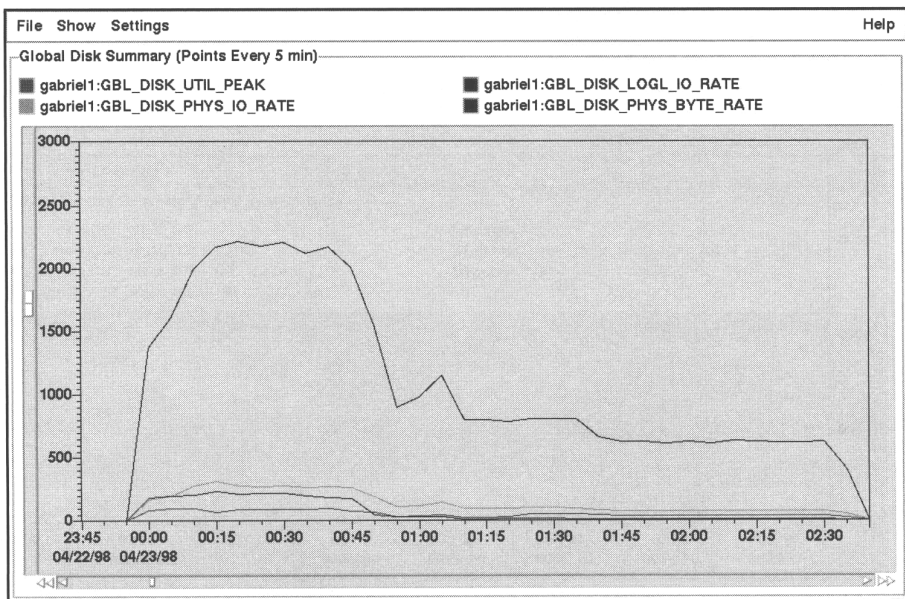



그림 14-14. 전역적디스크개요-짧은 시간창문

엄격히 3 시간에 대한 짧은 시간창문은 더 자세한 많은 정보를 보여 준다. 디스크능동성의 심한 뾰족점은 밤시간의 중간에서 나타난다. 그 리유로는 묶음적일감처리나 체계거꿀복사가 될수 있다.

한번에 한 체계자원에 관계되는 파라메터들만 보는것으로 제한받지는 않는다. 그림 14-15 를 통하여 많은 체계자원들이 동시에 리용되는가를 알수 있다.

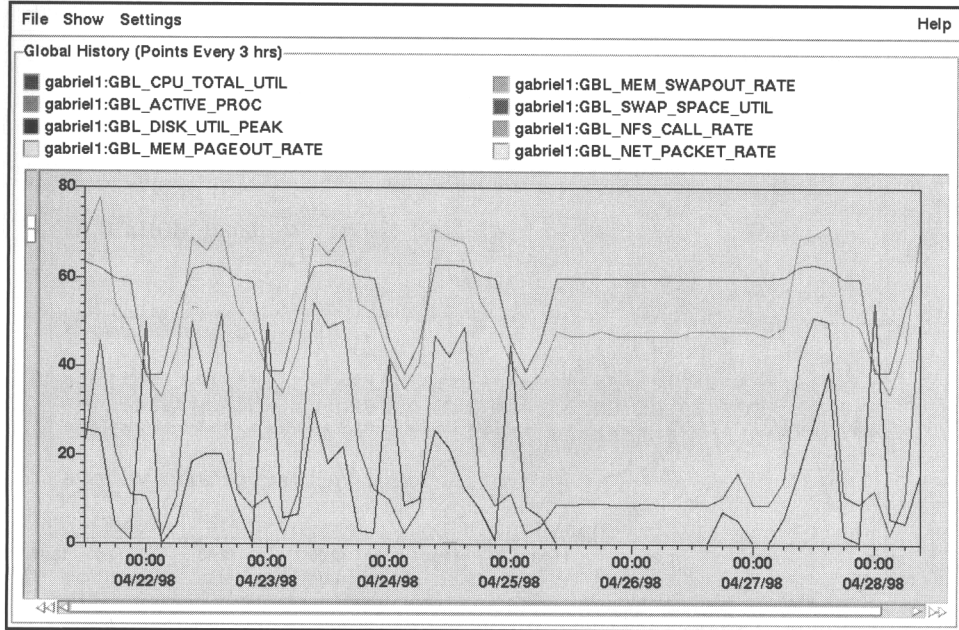



그림 14-15. 전역적개요리력화면

CPU, 디스크, 기억을 포함하여 많은 체계자원들이 이 그라프에서 제시되여 있다. 많은 체계자원들을 동시에 표시하였기때문에 더 짧은 많은 시간주기를 안전하게 볼수 있게 하고 있다.

그림 14-16 은 여러개의 짧은 시간창문을 통하여 같은 파라메터들을 보여 준다.

그림 14-16 은 더 짧은 시간창문으로 많은 체계자원을 리용할 때 더 명백한 립자모양을 볼수 있게 한다. 그림에서는 여러개의 체계자원들이 다른 체계자원들에 관계된다는 것을 볼수 있다.

편리한 지령 perfstat 에 의하여 체계에서 VantagePoint 수행해석기 /UX 의 상태를 볼수 있다.

다음실례는 perfstat 의 모든 선택항목들을 보기 위하여 지령 perfstat -?의 수행결과를 보여 준다.

perfstat -?

usage: perfstat [options]

Unix 선택항목	기능
-?	perfstat 의 모든 선택항목들을 목록화한다.
-c	체계의 모형통보를 보여 준다.
-e	수행도구의 상태파일들로부터 예고와 오유들의 찾기를 한다.
-f	수행도구의 상태파일들의 크기를 목록화한다.
-P	능동수행도구의 프로쎄스들을 목록화한다.
-t	수행도구의 상태파일들의 마지막일부 행들을 표시한다.
-v	수행도구의 파일들의 방안행들을 목록화한다.
-z	파일 혹은 띠에 perfstat 의 통보를 쏟는다.

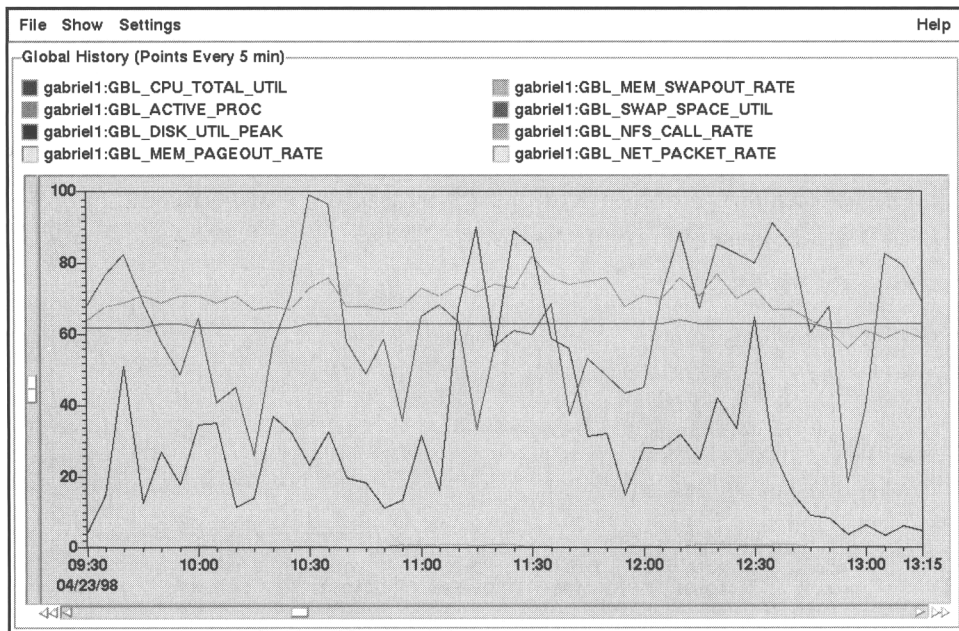



그림 14-16. 전역적개요 - 짧은 시간창문

다음 목록에서 보여 준것처럼 선택항목 -c 를 리용하면 체계모형에 대한 정보를 얻을수 있다.

```
# perfstat -c

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *    perfstat for rp-ux6 on Fri May 15 12:20:06 EDT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

system configuration information:

uname -a: HP-UX ux6 B.11.00 E 9000/800 71763 8-user license

mounted file systems with disk space shown:

Filesystem	kbytes	used	avail	%used	Mounted on
/dev/vg00/lvo13	86016	27675	54736	34%	/
/dev/vg00/lvo11	67733	44928	16031	74%	/stand
/dev/vg00/lvo18	163840	66995	90927	42%	/var
/dev/vg00/lvo17	499712	358775	132155	73%	/usr
/dev/rp06vgtmp/tmp	4319777	1099297	3134084	26%	/tmp
/dev/vg00/lvo16	270336	188902	7640S	71%	/opt
/dev/vgr00ti/var	640691	15636	605834	3%	/newvar
/dev/vgr00t1/usr	486677	356866	115210	76%	/newusr
/dev/vgr00t1/stand	67733	45109	15850	74%	/newstand
/dev/vgr00t1/root	83733	21181	54178	28%	/newroot
/dev/vgr00t1/opt	2632S3	188109	67246	74%	/newopt
/dev/vg00/lvol5	20480	1109	18168	6%	/home

LAN interfaces:

Name	Mtu	Network	Address	Ipkts	Opkts
100	4136	127.0.0.0	localhost	7442	7442
lan0	1500	192.60.11.0	rp-ux6	7847831	12939169

* * * * * * * * (end of perfstat -c output) * * * * * * * * * * * *

다음 목록에서 보여 준것처럼 선택항목 -f 를 리용하면 수행도구들의 상태파일의 크기를 알수 있다.

perfstat -f

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *    perfstat for ux6 on Fri May 15 12:20:08 EDT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

ls -l list of performance tool status files in /var/opt/perf:

```
-rw-rw-rw-  1  root  root  7812  May  10  19:35  status.alarmgen
-rw-r--r--  1  root  root     0  May  10  02:40  status.mi
-rw-rw-rw-  1  root  root  3100  May  10  02:40  status.perflbd
```

```
-rw-rw-rw-   1   root   root   3978   May   10   02:40   status.rep_server
-rw-r--r--   1   root   root   6079   May   11   23:30   status.scope
-rw-r--r--   1   root   root      0   May   31   07:26   status.ttd
* * * * * * * * (end of perfstat -f output) * * * * * * * * * * * *
```

다음 목록에서 보여 준것처럼 선택항목 -v 를 리용하면 수행도구띠수행의 방안의 행을 표시할수 있다.

perfstat -v

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *     perfstat for ux6 on Fri May 15 12:20:08 EDT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

수행도구파일들에 대한 방안행들을 목록화한다.

주의 : 다음의 쏘프트웨어방안통보들은 /opt/perf/ReleaseNotes 파일들에서 보여 준 방안통보와 비교될수 있다.

```
Measureware executables in the directory /opt /perf /bin
        scopeux   C. 01.00      12/17/97  HP-UX  11.0+
            ttd   A. 11.00.15   12/15/97  HP-UX  11.00
        perflbd   C. 01.00      12/17/97  HP-UX  11.0+
       alarmgen   C. 01.00      12/17/97  HP-UX  11.0+
     agdbserver   C. 01.00      12/17/97  HP-UX  11.0+
        agsysdb   C. 01.00      12/17/97  HP-UX  11.0+
     rep_server   C. 01.00      12/17/97  HP-UX  11.0+
        extract   C. 01.00      12/17/97  HP-UX  11.0+
        utility   C. 01.00      12/17/97  HP-UX  11.0+
            mwa   A. 10.52      12/05/97
       perfstat   A. 11.01      11/19/97
         dsilog   C. 01.00      12/17/97  HP-UX  11.0+
        sdlcomp   C. 01.00      12/17/97  HP-UX  11.0+
        sdlexpt   C. 01.00      12/17/97  HP-UX  11.0+
     sdlgendata   C. 01.00      12/17/97  HP-UX  11.0+
        sdlutil   C. 01.00      12/17/97  HP-UX  11.0+

Measureware libraries in the directory /opt /perf /lib
```

libmwa.sl C. 01.00 12/17/97 HP-UX 11.0+
libarm.a A. 11.00.15 12/15/97 HP-UX 11.00
liarm.sl A. 11.00.15 12/15/97 HP-UX 11.00

Measureware metric description file in the directory /var /opt /perf
metdesc C.01.00 12/17/97

All critical Measureware files are accessible
libnums.sl B. 11.00.15 12/15/97 HP-UX 11.00
midaemon B. 11.00.15 12/15/97 HP-UX 11.00
glance B. 11.01 12/16/97 HP-UX 11.00
gpm B. 11.01 12/16/97 HP-UX 11.00
* * * * * * * * (end of perfstat -v output) * * * * * * * * * * * *

지령소개

이 장에는 많은 지령들을 포함하고 있다. 앞의 실례들을 통하여 많은 지령들에 대한 간단한 설명을 주었다. 이 장에서 리용된 많은 지령들을 HP-UX 편람으로 제시한다. 편람에서는 전반적인 매 지령들에 대하여 더 상세하게 설명하였다.

iostat

iostat – I/O와 CPU통계자료의 호상작용에 대한 통보를 준다.

---

iostat(1)

이름

iostat – I/O통계자료를 통보한다.

형식

iostat [-t] [interval [count]]

해설

iostat는 체계에서 매 능동적디스크의 I/O 통계자료들의 호상작용에 대한 통보를 한다. 디스크자료는 4개 렬로 구성되여 있다. 즉

렬머리	설명
device	장치이름
bps	초당 전송된 KByte
sps	초당 찾기회수
msps	평균찾기당 미리초(ms)

만일 두개이상의 디스크가 있다면 자료는 매 디스크에 대하여 련속적으로 제출된다.
이 통보는 찾기, 자료전송완성, 전송된 단어수의 계산을 위하여 매 판에서 계산될수 있다. 또한 매 디스크의 상태는 초당 HZ번 검사된다. (sys/param.h에서 정의된 것과 같다.) 그리고 디스크가 능동이면 꼬리표가 만들어 진다. 이 자료안에서의 수들은 매 장치에서 평균찾기시간을 결정하기 위하여 전송비률과 결합될수 있다. 여러개의 디스크를 통하여 기입되는 자료해체와 같은 새 디스크기술의 출현으로 하여 단일한 자료전송은 평균 찾기당 미리초수를 정확히 계산하는데서 중요한것으로 된다. 여러가지 동적인 체계조건들에 의하여 이 값이 크게 변하므로 근사적인 표시만을 준다. 낮은 방안과 호환성을 보장하기 위하여 평균찾기당 ms(msps)마당을 1.0값으로 설정하였다.

선택항목들

iostat는 다음과 같은 선택항목들과 지령행의 인수들을 식별한다.

-t 디스크의 통계자료는 물론 말단 통계자료를 통보한다. 말단통계자료는 다음과 같은 마당이름들을 포함한다.

tin	말단으로부터 읽은 기호개수
tout	말단에 쓴 기호개수
us	체계가 사용자방식에서 보낸 시간의 퍼센트
ni	낮은 우선권(좋은)프로쎄스들을 수행하는 사용자방식에서 체계가 보낸 시간의 퍼센트
sy	체계방식에서 보낸 시간의 퍼센트
id	체계는 놀고 있는 상태에서 보낸 시간의 퍼센트

interval 초단위로 마지막시간구간의 총 합으로 되는 련속적인 행들을 표시한다. 첫 행은 재시동후에 시간에 대한 정보이고 나머지 행들은 마지막시간구간에 대한 정보들이다.

count 통계자료의 계산회수를 통보한다.

실례

모든 디스크들의 현재 I/O통계자료를 보려면

iostat

를 준다.

INTERRUPT 혹은 QUIT 를 누를 때까지 매번 10s 마다 모든 디스크의 I/O 통계자료를 표시하게 하려면

iostat 10

을 준다.

매번 10s 마다 모든 디스크의 I/O 통계자료를 표시하고 5 번 순차적읽기를 한후에 종결을 하려면

iostat 10 5

를 준다.

매번 10s 마다 모든 디스크의 I/O 통계자료를 표시하고 말단과 처리장치의 통계자료를 보고 5 번 련속적으로 읽기를 한후에 종결하려면

iostate -t 10 5

를 준다.

경고

정확한 마당너비와 출력령역은 체계, HP-UX 공개, 표시될 자료에 따라서 변할수 있다는것을 iostat 의 사용자는 알아야 한다.

저자

iostat 는 캘리포니아종합대학, 버클리, HP 에 의하여 개발되였다.

파일들

/usr/include/sys/param.h

관련항목

vmstat(1)

sar

sar - 체계의 능동성통보자이다.

---

sar(1M)

이름

sar - 체계의 능동성통보자이다.

형식

sar [-ubdycwaqvmAMS] [-o file] t [n]
sar [-ubdycwaqvmAMS] [-s time] [-e time] [-i sec] [-f file]

해설

우의 형식에서 보여 준 sar 는 t 초의 n 개 구간에서 조작체계가 조우한 능동프로쎄스들의 자료를 루적하면서 검사한다. 만일 선택항목 -o 가 규정되면 파일에 2 진형식으로 실험자료를 보관한다. n 의 기정값은 1 이다. 시간구간을 규정하지 않는 둘째 형식에서의 sar 는 이전에 기록된 파일로부터 자료를 읽는데 파일은 선택항목 -f 로 규정되거나 기정적으로는 표준체계의 능동적인 매일 자료파일인 /var/adm/sa/sadd 이다. 여기서 dd 는 현재 날자이다. 통보의 시작과 끝시간은 hh[:mm[:ss]]형식으로 시간인수 -s 와 -e 들로 규정된다. 선택항목 -i 는 sec 로 규정된 초구간의 기록들을 선택한다. 다른 경우에는 자료파일에서 찾아 진 모든 시간구간들을 통보한다.
어떤 경우나 인쇄된 자료의 부분모임은 다음의 선택항목들을 규정한다.

-u
CPU 리용을 통보(기정으로)한다. 즉 여러 방식들중의 한 방식에서 수행한 시간몫이다. 다중처리장치체계에서 만일 선택항목 -M 이 선택항목 -u 와 함께 리용된다면 모든 처리장치들의 평균 CPU 리용상태는 물론 CPU 당 리용상태도 통보된다. 만일 선택항목 -M 이 리용되지 않으면 모든 처리장치들의 평균 CPU 리용상태만 통보된다. 즉

- cpu CPU 번호(선택항목 −M 을 가진 다중처리장치체계에서만)
- %usr 사용자방식
- %sys 체계방식
- %wio I/O 대기에 의한 일부 프로쎄스들의 놀고 있는 상태(지적된 블로크 I/O, 미숙한 I/O, VM 의 페지입력/교환입력에 대하여서만)
- %idle 다른 경우 휴지시간

-b
완충기의 능동성을 통보한다.

- bread/s 디스크(혹은 블로크장치)로부터 고속완충기까지의 초당 물리적읽기회수
- bwrit/s 고속완충기로부터 디스크(혹은 다른 블로크장치)까지의 초당 물리적쓰기회수
- lread/s 고속완충기로부터 초당 읽기회수
- lwrit/s 고속완충기에 초당 쓰기회수
- %rcache 읽기요청에 대한 고속완충기의 명중비률. 실례로 1 은

bread/lread 이다.

%wcache 쓰기요청에 대한 고속완충기의 명중비률. 실례로 1 은 bwrit/lwrit 이다.

pread/s physio( )함수(서투른 I/O)에 의한 기호장치로부터 초당 읽기회수

pwrit/s physio( )함수(서투른 I/O)에 의한 기호장치에 초당 쓰기회수

-d 매 블로크장치에 대한 능동성을 통보한다.(실례로 디스크 혹은 디스크장치) 첫 행은 마지막구간에서 능동성을 가진 매 장치의 정보를 인쇄한다. 만일 능동성을 가진 장치가 없다면 빈 행이 인쇄된다. 매 행은 다음과 같은 마당을 포함한다.

device 장치의 론리적이름과 그것에 대응하는 구체례. 장치들은 다음과 같은 4 개 장치형으로 분류된다.

disk1 HP-IB 디스크(CS/80)

disk2 CIO HP-FL 디스크(CS/80)

disk3 SCSI 와 NID FL 디스크

sdisk SCSI 디스크

%busy 요청에 봉사한 장치의 시간부분(몫)

avgue 장치에서 미해결된 요청들에 대한 평균수

r+w/s 장치로부터 혹은 장치에로의 초당 자료전송회수(읽기와 쓰기)

blks/s 장치로부터 혹은 장치에로의 전송된 바이트수(512byte 단위로)

avwait 전송요청이 장치순서에서 대기한 평균시간(ms 로)

avserv 장치에 매 전송요청의 봉사에 대한 평균시간(찾기, 회전에 의한 지체, 자료전송시간을 포함한 ms 로)

-y tty 장치의 능동성을 통보한다.

rawch/s 초당 미숙한 입력기호들

canch/s canon()으로 처리된 초당 입력기호들

outch/s 초당 출력한 기호들

rcvin/s 초당 들어 온 기호중단

xmtin/s 초당 나간 기호중단

mdmin/s 모뎀중단비률(봉사하지 않으면 항상 0 이다.)

-c 체계호출을 통보한다.

scall/s 초당 모든 형태들의 체계호출의 회수

sread/s　초당 read( )와 readv( )체계호출의 회수
swrit/s　초당 write( )와 writev( )체계호출의 회수
fork/s　초당 fork( )와 vfork( )체계호출의 회수
exec/s　초당 exec( )체계호출의 회수
rchar/s　초당 읽기체계호출(블로크장치만)로 전송된 기호의 개수
wchar/s　초당 쓰기체계호출(블로크장치만)로 전송된 기호의 개수

-w　체계교환과 능동성절환을 통보한다.
swpin/s　초당 교환되여 들어 온 프로쎄스의 개수
swpot/s　초당 교환되여 나간 프로쎄스의 개수
bswin/s　초당 교환프로쎄스에 입력된 512Byte 단위로서의 개수
bswot/s　초당 교환프로쎄스에 출력된 512Byte 단위로서의 개수
pswch/s　초당 프로쎄스의 문맥이 절환된 회수

-a　파일입장체계루틴의 리용을 통보한다.
iget/s　초당 파일체계 iget( )호출의 회수
namei/s　초당 파일체계 lookuppn( )호출(경로이름 변환)의 회수
dirblk/s　초당 등록부표시를 목적으로 파일체계의 블로크를 읽은 회수

-q　봉사될동안 평균순서길이와 봉사된 시간의 퍼센트를 통보한다. 만일 선택항목 −M 이 선택항목 -q 와 함께 리용된다면 다중처리장치체계에서 모든 처리장치의 평균수행순서는 물론 CPU 당 수행순서를 통보한다. 만일 선택항목 −M 이 리용되지 않으면 모든 처리장치의 평균 수행순서정보만 통보한다.
cpu　cpu 의 개수(다중처리장치체계에서만 선택항목 -M 과 함께 리용되는)
rung-sz　프로쎄스(기억에서 또는 수행할수 있는)의 수행순서의 평균길이
%runocc　수행순서가 프로쎄스(기억에서 또는 수행할수 있는)으로 봉사되는 시간의 퍼센트
swpq-sz　수행할수 있는 프로쎄스(프로쎄스는 교환출력되나 수행에 준비된)들이 교환순서에 봉사된 시간의 퍼센트
$swpocc　수행할수 있는 프로쎄스(프로쎄스들은 교환출력되나 수행에 준비된)들이 교환순서에 봉사된 시간들의 퍼센트

-v　본문, 프로쎄스, 내부연산과 파일표들의 상태를 통보한다.
text-sz　(적용할수 없다.)
proc-sz　프로쎄스표의 현재의 크기와 최대크기
inod-sz　내부연산표(내부연산고속완충기)의 현재의 크기와 최대크기

file-sz　체계파일표의 현재의 크기와 최대크기
text-ov　(적용할수 없다.)
proc-ov　검사할 점들사이에 프로쎄스표가 초과된 회수(핵심부가 임의로 리용할수 있는 프로쎄스표등록항목을 찾지 못한 회수)
inod-ov　검사할 점들사이에 내부연산표(내부연산고속완충기)가 초과된 회수(핵심부가 임의로 리용할수 있는 내부연산표의 등록항목을 찾지 못한 회수)
filde-ov　검사할 점들사이에 체계파일표가 초과된 회수(핵심부가 임의로 리용할수 있는 파일표등록항목을 찾기 위한 회수)

-m　통보와 기발의 능동성을 통보한다.
msg/s　초당 System V 의 msgrcv( )호출회수
sema/s　초당 System V 의 semop( )호출회수
select/s　초당 System V 의 select( )호출의 개수. 만일 선택항목 -S 가 명백히 규정되였다면 이 값은 그때에만 통보한다.

-A　모든 자료를 통보한다.
-udqbwcayvm 과 같다.

-M　선택항목 -q 와 −u 가 함께 리용될 때 다중처리장치체계에서 처리장치들의 자료를 통보한다. 만일 선택항목 −M 이 다중처리장치체계에서 리용되지 않았다면 선택항목 -u 와 -q 의 출력형식은 uni 처리장치의 출력형식과 같다. 통보된 자료는 모든 처리장치의 평균값과 같다.

실례

5s 동안에 CPU 의 능동성발생을 보자면

sar 1 5

로 준다.

10min 동안에 CPU 의 능동성발생을 보고 자료를 보관하자면

sar -o temp 60 10

으로 준다.

그 시간주기로부터 후에 디스크와 테프의 능동성을 다시 보자면

sar −d −f temp

로 준다.

다중처리장치체계에서 후에 CPU 의 리용을 다시 보자면

sar −u −M −f temp

로 준다.

경고

정확한 마당너비와 그 출력의 령역은 체계, HP-UX 공개, 표시될 자료에 의존하여 변한다는것을 sar의 사용자는 알고 있어야 한다.

파일들

/var/adm/sa/sadd 매일의 자료파일. 여기서 dd는 월에서의 날자를 표시하는 두 개의 수자이다.

관련항목

sal(1M)

표준일치

sar: SVID2, SVID3

showmount

showmount − 모든 원격장비를 보여 준다.

---

showmount(1M)

이름

showmount − 모든 원격장비를 보여 준다.

형식

/usr/sbin/showmount [-a] [-d] [-e] [host]

해설

showmount는 호스트로부터 파일체계를 원격적으로 설치한 모든 의뢰기들을 목록화한다. 이 정보는 호스트에서 봉사기를 설치하는것으로써 얻어 진다. 호스트의 기정값은 hostname으로 귀환된 값이다.

선택항목들

-a 다음의 형식으로 모든 원격적설치에 대한 통보를 인쇄한다.

이름 : 등록부

여기서 hostname은 의뢰기의 이름이고 등록부는 설치된 파일체계의 등록부 혹은 뿌리등록부이다.

-d 의뢰기에게 원격적으로 설치된 등록부들을 목록화한다.

-e **반출**(Export)된 파일체계의 목록을 인쇄한다.

경고

만일 의뢰기가 없어 진다면 봉사기에서 수행되는 showmount 지령은 의뢰기가 아직 설치된 파일체계를 가지고 있다는것을 보여 준다. 달리 말하여 의뢰기의 등록항목은 의뢰기가 재시동되고 지령 umount -a 를 수행할 때까지 /etc/rmtab 는 원격화되지 않는다.

만일 의뢰기가 같은 원격등록부를 두번이상 설치한다면 한 등록항목만 /etc/rmtab 에 나타난다. 이 등록부들중에서 한개 등록부를 설치하는것으로써 한 등록항목을 제거하면 showmount 는 원격등록부가 설치되였다는것을 더는 통보해주지 않는다.

저자

showmount 는 Sun Microsystems 연구소에서 개발하였다.

관련항목

hostname(1), exportfs(1M), mountd(1M), exports(4), rmtab(4)

swapinfo

swapinfo - 체계의 페지화의 정보를 통보한다.

---

swapinfo(1M)

이름

swapinfo - 체계페지화공간의 정보를 통보한다.

형식

/usr/sbin/swapinfo [-mtadfnrMqw]

해설

swapinfo 는 장치와 파일체계의 페지화공간에 대한 정보를 인쇄한다. (주의 : 술어 swap 는 가상기억을 실현하는데서 항상 필요하다. HP-UX 는 실제적으로 교환이라기보다 페지화에 의하여 가상기억을 실현한다. 이 지령과 다른 지령들은 swap 로부터 유도된 이름을 그대로 리용한다.)

기정으로 swapinfo 는 여기서 보여 준것처럼 페지령역당 한 행씩 놓이는 두개 행의 머리부를 표준출력에로 인쇄한다. 즉

Kb Kb Kb PCT START/ Kb

TYPE AVAIL USED FREE USED LIMIT RESERVE PRI NAME

마당들은 다음과 같다.

TYPE 다음과 같은 값들중의 하나이다.

dev 대용량기억장치에 상주하는 페지화공간은 장치기억령역전체를 차지하거나 만일 장치가 파일체계를 포함한다면 파일체계의 끝과 장치끝사이의 공간에 이 페지화공간이 차지하고 있다. 이 공간은 페지화를 위하여 독점적으로 리용되는데 만일 페지화를 리용하지 않는다고 하여도 이 공간을 다른 목적에 리용할수 없다. 장치의 페지화령역은 가장 빠른 페지화를 할수 있게 한다.

fs 파일체계로부터 리용할수 있는 동적인 페지화공간을 표시한다. 이 공간이 요구될 때 체계는 파일체계의 파일들을 배치하고 페지화공간처럼 그 파일을 리용한다. 파일체계의 페지화는 장치페지화보다 보통 느리다. 페지화가 요구되지 않을 때 다른 일(례컨대 사용자파일의 배치)에 이 공간을 리용할수 있게 한다.

localfs 국부적디스크에 상주하는 파일체계에서 파일체계페지화공간을 통보한다(우의 fs를 보기 바란다.).

network 다른 콤퓨터에 상주하는 파일체계에서 파일체계페지화공간에 대하여 통보한다(우의 fs를 보기 바란다.). 이 파일체계는 NFS를 통하여 국부적콤퓨터에 설치되여야 한다.

reserve 예정된 페지화공간을 통보한다. 이것은 현재 수행하고 있는 프로쎄스들에 필요되는 페지화공간의 총 용량이지만 우의 페지화령역들가운데서 그 어떤 령역에도 아직 배치되지 않은것이다.

memory 기억페지화령역을 통보한다(또한 허위교환으로 알려져 있다.). 이것은 우의 페지화령역들이 모두 리용되는 조건에서 페지들의 보관에 리용될수 있는 체계기억의 총량이다. 아래의 페지화배치를 보기 바란다. 이 행은 기억페지화가 가능할 때에만 나타난다.

Kb AVAIL 페지화령역에서 1024Byte의 블로크의 단위로 리용할수 있는 총 기억공간을 통보한다(필요하면 가장 가까운 블로크단위로

반올림한다.). 또한 이미 리용중에 있는 임의의 페지화공간들도 포함한다. 파일체계페지화령역에 대하여 이 값은 언제나 상수로 되지는 않는다. 그것은 페지화(현재 리용되지 않을 때까지도)에 배치된 현재의 공간의 크기와 파일체계에서 기본사용자들에게 리용할수 있는 자유블로크들의 크기를 더한데서 값 RESERVE 를 던 값이다(이 값은 0 보다 작을수는 없다.). 만일 LIMIT 가 령이 아니면 AVAIL 은 LIMIT 보다 더 클수 없다. 페지화공간은 큰 토막들에 배치되므로 AVAIL 은 가장 가까운 옹근수개수의 배치토막단위로 취한다.
기억페지화령역에 대하여 이 값은 언제나 상수로 되지는 않는다. 왜냐하면 페지화에 필요되는 프로쎄스들에는 물론 핵심부에 의하여 기억의 배치도 반영하기때문이다.

Kb USED　페지화령역에서 페지화에 리용된 1KByte 블로크단위로 현재개수를 통보한다. 기억페지화령역에 대하여 이 값은 다른 목적으로 리용된 기억을 포함하므로 페지화에 리용할수 없는 기억도 포함한다.

Kb FREE　앞으로 페지화에 리용될수 있는 공간의 총량을 표시한다. 보통 이것은 Kb AVAIL 과 Kb USED 의 차와 같다. 만일 장치페지화형식의 어떤 부분을 리용할수 없다면 페지화령역의 크기는 차이나는데 그 리유는 배치토막크기의 배수가 아니기때문에 혹은 상태의 파라메터 maxswapchu-nks 가 충분히 크게 설정되지 못하였기때문이다.

PCT USED　Kb USED 를 Kb AVAIL 로 나눈데 기초하여 리용하고 있는 용량의 퍼센트를 통보한다. 만일 Kb AVAIL 이 0 이면 100%로 본다.

START/LIMIT

장치페지화령역에 대하여 START 는 대용량기억장치에서 페지화령역의 시작블로크주소이다. 이 값은 페지화에 제공된 장치에 대하여 표준값은 0 이거나 파일체계와 페지화공간을 다 포함하는 장치들에 대하여 파일체계의 끝주소이다.
파일체계페지화령역에 대하여 LIMIT 는 교환할수 있는 최소값단위로서 페지화에 리용된 1KByte 블로크의 최대개수이다. 아무런 의미도 가지지 않는 파일체계의 LIMIT 값은 한계가 고정되지 않는다. 모든 공간은 minfree + RESERVE 로 반영된 크기보다 작은 파일들은 이 모든 공간을 자유로 리용할수 있다.

RESERVE　장치페지령역에 대하여 이 값은 항상 "-"이다. 파일체계페지화

령역에 대하여 이 값은 교환에 주어 진 예정값과 같은 기본사용자에 의하여 리용하는 파일체계에 예정된 1KByte 블로크의 개수이다.

PRI　　교환하는데서 주어 진 우선권값과 같다. 이 값은 페지화에 리용된 장치들과 파일체계들에 대하여 기억공간을 배치받을 순서를 규정해 준다. 가장 낮은 우선권값을 가진 령역에서 취한 공간은 첫 순서로 배치된다. 우선권은 0 과 10 사이의 값을 가진다. 아래에 Paging Allocation 을 참고.

NAME　　장치페지화령역에 대하여 기본번호와 보조번호가 장치의 ID 와 일치하는 전문적파일이름의 블로크이다. swapinfo 지령은 장치이름을 찾기 위하여 /dev 나무를 찾는다. 만일 일치하는 전문적파일블로크가 찾아 지지 않으면 swapinfo 는 장치 ID 를 출력 (기본번호와 보조번호의 값) 한다. 실례로 28, 0X15000 이다.

파일체계교환령역에 대하여 NAME 은 페지화파일들이 기억된 파일체계에서의 등록부의 이름이다.

Paging Allocation(페지화배치)

페지화령역들은 체계를 시동시킬 때(핵심부에서 모형화된 장치페지화령역에 대하여) 조성할수 있는데 /etc/fstab 의 내용에 기초하여 체계를 시동시킬 때 /sbin/init.d/swap_satrt 에 의하여 보통 기동되는 swapon 지령의 리용으로 조성될수 있다. 페지화령역이 리용될수 있을 때 그 령역의 일부분은 페지화공간에 배치된다. 장치페지화령역에 대하여 완전장치(entire device)는 어떤 배치토막의 임의의 나머지 부분보다 작게 배치된다(한 배치토막의 크기는 상태파라메터 swchunk 로 조종되는데 보통 2MByte 이다.). 파일체계페지화령역에 대하여 swapon 에서 주어진 최소값은 배치(가장 가까운 배치토막까지 반올림되며)된다.

프로쎄스가 조성될 때 혹은 추가적공간이 요구될 때 페지화공간은 우의 예정행에서 보여 준 공간을 증가시키며 그것을 이 공간에 포함시킨다. 페지화능동성이 실지로 나타날 때 공간은 페지화령역들중 한 령역(이미 배치되고 리용할수 있는 자유공간을 가지는 가장 낮은 우선권번호를 가진 페지화령역)에서 리용되고 그 공간은 그 령역에서 리용된것과 같이 표시된다.
모든 페지화령역에서 리용된 공간의 합과 예정된 공간의 총량의 합은 모든 페지화령역들이 배치된 총 용량을 초과할수는 없다. 만일 더 많은 기억요청이 일어난다면 체계는 아래에서 제시된 방안을 선택하여야 한다. 즉

i. 체계는 파일체계페지화령역들에 더 많은 공간배치로 리용할수 있는 총 공간을 증가시키려고 한다.

ii. 만일 모든 파일체계페지화령역들이 완전히 배치되고 요청을 아직 만족시키지 않으면 체계는 우의 기억행에서 서술한것처럼 기억페지화리용을 하려고 한다(기억페지화는 기정값으로 1(on)로 되는 상태의 파라메터 swapmem_on 으로 조종된다.). 만일 이 파라메터가 0(off)으로 되면 기억행은 나타나지 않을것이다.

iii. 만일 기억페지화가 요청을 만족시킬수 없다면 완전히 배치되였거나 0(off)으로 되였기때문에 요청은 거절된다.

이 방안의 여러가지 실현들이 있기때문에 swapinfo 의 출력의 결과를 잘 리해하자면 주의하여 보아야 한다.

- 페지화공간은 파일체계페지화령역이 더 낮은 우선권값을 가지지만 모든 장치페지화공간이 예정되기전에는 그 파일체계페지화령역에서 배치되지않을것(령역을 리용할수 있고 최소의 첫 령역을 제외하고)이다.
- 사용자파일들은 페지화공간이 파일체계페지화령역에 배치되지만 페지화능동성이 없을 때에는 그 공간을 리용할수 없게 된다.
- 더 많은 페지화공간의 요청은 예정장치, 파일체계, 기억페지화에 의하여 만족될수 없을 때에는 물론이고 예정된 페지화공간의 일부가 아직 리용되지 않을 때에도 실패한다. 이처럼 더 많은 페지화공간의 요청은 페지화령역의 일부 혹은 전체가 리용되지 않았다는것을 보여 주고 있으며 이 령역들에서 공간이 완전히 예정되였을 때에 거절될수 있다는것을 보여 주고 있다.
- 체계가 리용할수 있는 기억은 페지화부분체계와 핵심부기억배치의 부분으로 구분된다. 그리하여 체계는 리용할수 있는 모든 디스크페지화공간이 완전히 예정되면 그 체계가 완전히 배치될 때까지 기억페지화를 리용할수 있다.

선택항목들

swapinfo 는 다음의 선택항목들을 식별한다.

-m KByte 대신에 가장 가까운 완전 MByte(1024^2 의 배수)단위를 리용한다. 자료는 AVAIL, USED, FREE, LIMIT 와 RESERVE 값을 MByte 단위로 표시한다. 출력의 머리부형식은 각각 마당이름 Kb 를 마당이름 Mb 로 변화시킨다.

-t 마지막으로 TYPE 에 의하여 총계를 주는 행을 추가한다. 이 행은 모든 페지화령역들이 아니라 그 우에 표시된 페지화정보들만의 총합이다. 만일 -dfrM 의 부분모임이 규정된다면 이 행은 오해를 일으킬수 있다.

-a 핵심부로 모형화되지만 현재 불가능한 장치페지화령역들을 포함하여 그 모든 장치페지화령역들을 보여 준다(이 선택항목을 주지 않으면 이런 통보는 표시되지 않는다). 마당이름 NAME 뒤에 불가능하다는 단어가

나타나면 Kb AVAIL, Kb USED, Kb FREE 들의 값은 각각 0 이다. 선택항목 -a 는 선택항목 -d 가 제출되거나 기정으로 true 로 되면 무시된다.

-d 장치페지화령역의 정보만을 인쇄한다. 이것은 출력머리부를 변경시킨다.

-f 파일체계페지화령역의 정보만을 인쇄한다. 이것은 출력머리부를 적당히 변경시킨다.

-n fs 령역을 호출할 대신에 localfs 과 network 의 두개 령역으로 파일체계페지화령역을 분류한다.

-r 예정된 페지화공간의 정보만을 인쇄한다.

-M 기억페지화령역의 정보만을 인쇄한다. 선택항목 -d, -f, -n, -r, -M 들은 결합될수 있다. 기정으로는 -dfnrM 이다.

-q 끝내기방식이다. 마당 Kb AVAIL 값만을 인쇄한다(선택항목 -m 은 마당 Mb AVAIL 에로 출력된다.). 즉 빨리 총합을 요구하는 프로그람들이 리용할수 있게 하기 위한 체계에서 리용할수 있는 총 페지화령역(즉 선택항목 -d, -f, -r, -M 들이 규정될 때에만 장치, 파일체계, 예정, 기억페지화공간)이다. 만일 -q 가 규정되면 선택항목 -t 와 -a 들은 무시된다.

-w 쓸모 없는 공간(즉 배치된 크기가 그 페지화령역의 총 크기보다 적은 임의의 장치페지화령역)을 포함하는 매 장치페지령역에 대한 경고를 인쇄한다. 이 선택항목은 -d 로 규정되거나 기정으로 true 일 때에만 작용한다.

귀환값

swapinfo 는 만일 완전히 성공(임의의 경고가 출력된것을 포함하여)하면 0, 임의의 오유통보가 표시되면 1 을 귀환한다.

진단

swapinfo 는 만일 어떤 문제라도 발생하면 표준오유에 대한 정보를 인쇄한다.

실례

총 합계를 표시하는 행을 가진 모든 파일체계페지화령역을 목록화하려면 다음과 같이 리용할수 있다.

```
swapinfo −ft
```

경고

swapinfo 는 핵심부의 일부 정보에로 접근할것을 요구한다. 만일 사용자가 핵심부접근에로의 적당한 특정의 수준을 가지지 못한다면 swapinfo 는 경고통보를 내보내는데 그 정보의 기정값은 변경되지 않는다고 간주한다.

swapinfo 의 사용자들은 그 출력의 령역과 정확한 마당너비가 체계, HP-UX 공개, 표시될 자료에 의존하여 변할수 있다는것을 알아야 한다.

페지화배치를 달리 실현할 때 이 편람의 정보는 경고없이 변경될수 있다. 사용

자들은 이 정보의 정확성을 확인하고 리용하여야 한다.

저자

swapinfo 는 HP 에 의하여 개발되였다.

관련항목

swapon(1M), swapon(2), fstab(4), fs(4)

timex

timex – 어떤 지령의 수행시간을 측정하고 체계능동성의 통보를 표시한다.

---

timex(1)

이름

timex – 어떤 지령의 수행시간을 측정한다. 프로쎄스의 자료와 체계의 능동성을 통보한다.

형식

timex [-o] [-p[fhkmrt]] [-s] command

해설

timex 는 초단위로 지나간 시간, 사용자시간, 주어 진 지령의 수행으로 보낸 체계시간을 통보한다. 선택적으로 이 지령의 자료를 리용하는 프로쎄스들과 그 프로쎄스의 모든 후손프로쎄스들은 목록화되거나 개괄되고 수행구간에서 총체적인 체계능동성을 통보한다.

timex 의 출력은 표준오유출력장치에로 출력된다.

선택항목들

-o 읽거나 쓴 블로크의 총수와 지령과 그 지령의 모든 후손프로쎄스에 의하여 전송된 총 기호들을 통보한다.

-p[fhkmrt] 지령과 그것의 모든 후손프로쎄스의 기록들을 리용하는 프로쎄스들을 목록화한다. 부분선택항목 f, h, k, m, r, t 들은 통보되는 자료항목들을 변경시킨다. 그것들은 지령 acctcom(1M)에서 정의된 선택항목들과 같다. 읽거나 쓴 블로크의 수와 전송된 기호개수는 항상 통보된다.

-s 지령의 수행기간에 나타난 총 체계능동성(지령과 관련되지 않는)을 통보한다. sar(1)에서 목록화된 모든 자료항목들은 통보된다.

실례

단순한 실례로서

```
timex -ops sleep 60
```

이다.

임의의 복잡성을 가진 말단기간은 부분쉘의 시간화로 측정할수 있다. 즉

```
timex -opskmt sh
        session commands
EOT
```

경고

프로쎄스의 혈통이 리용될수 없으므로 지령과 관련된 프로쎄스의 기록들은 계산서파일 /var/adm/pacct 로부터 참조하여 선택된다. 동일한 사용자 ID, 말단 ID 를 가지는 배경프로쎄스들과 수행시간창문은 가상적으로 나타난다.

관련항목

sar(1), acctwm(1M)

표준일치

timex:SVID2, SVID3

top

top - 체계에서 꼭대기프로쎄스들에 대한 정보를 제공한다.

---

top(1)

이름

top - 체계에서 꼭대기프로쎄스들에 대한 정보를 표시하고 갱신한다.

형식

top [-s time] [-d count] [-q] [-u] [-n number]

해설

top 는 체계에서 꼭대기프로쎄스들을 표시하고 주기적으로 정보를 갱신한다. 이 지령은 미숙한 CPU 리용퍼센트가 프로쎄스들을 정리하는데도 리용된다.

선택항목들

top는 다음과 같은 지령행의 선택항목들을 식별한다.

-s time	시간 time 초마다 화면이 갱신될 지체시간을 설정한다. 화면이 갱신될 기정지체시간은 5초이다.
-d count	count 회수만큼 표시를 하고 종결한다. 여기서 표시는 화면이 한번 바꾸어 지는것을 의미한다. 이 선택항목은 프로그람이 종결전에 보여 줄 표시의 회수를 규정하는데 리용된다.
-q	이 선택항목은 지령 nice -20 의 수행과 같은 우선권에서 꼭대기프로그람을 더 빨리 수행하도록 한다. 이 선택항목은 체계가 대단히 느린 경우에 임의의 체계에서 제기된 문제를 발견하는데 유리할수 있다. 이 선택항목은 적당한 특정권한을 가지는 사용자만이 사용할수 있다.
-u	사용자이름대신에 사용자 ID(uid)번호를 표시하게 한다. 이 기능은 uid 번호를 사용자이름으로 대응시키는데 필요한 추가적시간을 없애므로 수행시간을 줄이게 한다.
-n number	매 화면에 number 개의 프로쎄스들만 표시한다. 이 개수가 매 화면에 표시될수 있는 프로쎄스의 최대수보다 크다면 이 선택항목은 무시된다.

화면조종지령들

다중화면자료를 표시할 때 꼭대기에는 다음과 같은 화면건반조종지령들이 표시된다.

j	만일 현재 화면이 마지막화면이 아니라면 다음 화면을 표시한다.
k	만일 현재 화면이 첫 화면이 아니라면 이전 화면을 표시한다.
t	첫(꼭대기) 화면을 표시한다.

프로그람종결

프로그람을 종결하고 표준사용자프로쎄스에로 다시 가려면 임의의 시간에 q 를 입력하면 된다.

출력통보에 대한 설명

정보는 3개의 일반적클라스로 꼭대기에 표시된다.

System Data:

표시의 꼭대기에서 첫 일부 행들은 다음과 같은 체계의 일반상태정보를 보여 준다.

- 체계이름과 현재시간
- 마지막 1, 5, 15 분에서 평균부하

· 존재하는 프로쎄스의 개수와 매 상태들 즉 잠자기 (sleep), 대기(wait), 수행(run), 시작(start), 갱신 (zombie), 정지(stop)에서 프로쎄스의 개수
· 체계의 매 처리장치의 상태(사용자, 좋은, 체계, 놀기, 중단과 교환자)에서 보낸 시간의 퍼센트
· 매개 처리장치상태의 평균값(다중처리장치체계에서만)

Memory Data

리용하고 있는 가상기억 및 실제기억(능동성으로 고려된 기억용량)과 자유기억의 총 용량을 포함한다.

Process Data

체계에서 개별적인 프로쎄스들에 대한 정보를 제시한다. 프로쎄스들의 자료가 한개 화면에 표시할수 없을 때 꼭대기는 두개이상의 화면들로 자료를 분할한다. 다중화면자료를 보려면 이미 서술된 지령 j, k, t들을 리용하여야 한다.

체계자료 및 기억자료표시는 다중화면의 프로쎄스자료와 함께 매 화면에 제출된다. 프로쎄스자료는 지령 ps(1)을 리용할 때의 출력자료와 류사한 형식으로 다음과 같이 표시된다.

CPU	프로쎄스(다중처리장치체계에서만)를 수행하는 처리장치번호
TTY	프로쎄스에서 리용된 말단대면부
PID	프로쎄스 ID의 번호
USERNAME	프로쎄스의 소유자이름. 선택항목 -u가 규정될 때 사용자 ID(uid)는 USERNAME대신에 표시된다.
PRI	프로쎄스의 현재 우선권
NI	-20으로부터 +20까지 범위의 값
SIZE	KByte로 표시된 프로쎄스의 총 크기. 이것은 본문, 자료, 탄창을 포함한다.
RES	KByte로 표시된 프로쎄스의 상주부분에 대한 크기. 이 상주부분의 크기정보는 근사적값이다.
STATE	프로쎄스의 현재상태. 여러가지 상태는 잠자기, 대기, 수행, 놀기, 갱신, 정지와 같은것이다.
TIME	프로쎄스가 소비한 체계와 CPU의 초단위로 표시된 시간값
%WCPU	큰 부하를 받는 CPU(중앙처리장치)의 퍼센트
%CPU	미숙한 CPU 퍼센트. 이 마당은 꼭대기프로쎄스를 정돈하는데 리용된다.
COMMAND	현재 수행되는 프로쎄스의 지령이름

실례

top 는 지령행의 선택항목을 가지고 혹은 없이 수행될수 있다. 2 초간격으로 5 개 자료화면을 표시하고 자동적으로 종결하려면 다음의 지령을 주면 된다.

top −s2 −d5

저자

HP 와 라이스종합대학의 윌리암 레피버에 의하여 개발되였다.

vmstat

vmstat − 프로쎄스, 가상기억, 함정, CPU 의 능동화를 통보한다.

---

vmstat(1)

이름

vmstat − 가상기억의 통계자료를 통보한다.

형식

vmstat [-dnS] [interval] [count]
vmstat −f | -s | -2

해설

지령 vmstat 는 프로쎄스들, 가상기억, 함정, CPU 능동성에 대한 일정한 통계자료의 보존상태를 통보한다. 이것은 핵심부의 기본구조에서 가산기들을 지울수 있다.

선택항목들

vmstat 지령은 다음과 같은 선택항목들을 식별한다.

-d 개별적부분(separate section)과 같이 디스크전송정보를 초당 전송하는 형식으로 통보한다.

-n 80 개렬로 구성된 표시장치에서 보기 편리한 출력형식을 규정한다. 이 형식은 가상기억정보와 CPU 자료인 두 그룹으로 표준출력을 구분한다. 매 그룹은 출력에서 개별적인 한 행으로 표시된다. 다중처리장치체계에서 이 표시형식은 주로 CPU 별로 리용하게 한다.

-S 페지개선과 주소변환실패(re 와 at)한 개수를 통보하는것이 아니라 입력과 출력(si 와 so)으로 교환된 프로쎄스의 개수를 통보한다.

interval 마지막 interval 초동안 개괄되는 성공적행들을 표시한다. 만일 interval 이 령이면 출력은 한번씩 표시된다. 만일 선택항목 -d 가 규정되면 렬

의 머리는 반복된다. 만일 -d 가 생략되면 렬의 머리는 반복되지 않는다. 지령 vmstat 5 에 의하여 체계는 매 5 초마다 정보를 인쇄한다. 이 지령은 체계에서 검사될 일부 통계자료만 출력하기때문에 인쇄구간을 선택하는데서 좋은 방법으로 된다. 다른 경우에는 매 초마다 변한다.

count　개괄통계자료를 count 번 반복한다. 만일 count 가 생략되거나 0 이면 출력은 중단 혹은 종결신호를 받을 때까지 반복한다. 이 신호들은 말단으로부터 입력할 때 공통적으로 각각 ^C 와 ^\이다.

-f　갈래의 개수와 체계시동된후에 기동된 가상기억의 페지수를 통보한다.

-s　체계시동된후에 혹은 지령 vmstat -z 가 마지막으로 수행된 후에 나타난 핵심부의 기본구조로부터 페지화와 관련된 모든 사건들의 총수를 인쇄한다.

-z　핵심부의 기본구조에서 모든 가산기들을 지운다. 이것은 /dex/kmem 에 대한 파일쓰기입장허용성을 요구한다. 이것은 적당한 특정수준을 가진 사용자에게만 표준적으로 허용된다.

만일 이 선택항목들이 주어 지지 않는다면 vmstat 는 체계시동된 후에 혹은 지령 vmstat -z 가 마지막으로 수행된 후에 가상기억의 능동성에 대한 통보를 한행에 표시한다.

렬에 대한 해설

렬머리부와 매 렬의 의미는 다음과 같다.

procs　각이한 상태에서 프로쎄스의 개수에 대한 정보이다.

r　수행순서이다.
b　자원에 대한 봉쇄(I/O, 페지화들) 정보이다.
w　수행할수 있거나 짧은 잠자기(<20 초)이지만 교환된다.

memory　가상기억 및 실제기억의 리용에 대한 정보. 만일 가상페지들이 마지막 20 초동안에 수행되고 있거나 수행된 프로쎄스에 속한다면 능동적인것으로 간주한다.

avm　가상페지들의 능동화.
free　자유목록의 크기.

page　페지실패의 페지화능동성의 통보. 이 통보는 초단위들로 주어 지는데 초단위는 매 5 초구간에서 평균값으로 정의된다.

re　페지개선(-S 없는)
at　주소변환실패(-S 없는)
si　안으로 교환된 프로쎄스(-S 가진)
so　밖으로 교환된 프로쎄스(-S 가진)

pi 들어 온 페지들
po 나간 페지들
fr 초당 자유로운 페지들
de 예견된 짧은 기한의 기억충돌
sr 초당 시계알고리듬으로 조사된 페지들

faults 마지막 5s 동안 초당 함정/중단 평균비률

in 초당 장치중단들(시계중단제외)
sy 초당 체계호출
cs CPU 문맥절환비률(절환회수/초)

cpu cpu 시간리용의 고장퍼센트

us 표준과 낮은 우선권프로쎄스들의 사용자시간
dy 체계시간
id CPU 놀기

실례

다음 실례들은 여러개 지령선택항목들을 리용한 출력결과에 대하여 보여 준다. 형식화를 목적으로 일부 공백들은 제거되였다.

① 기정출력을 표시한다.

```
vmstat
   procs   memory             page                        faults       cpu
r  b  w  avm   free  re  at  pi  po  fr  de  sr   in  sy  cs  us  sy   id
0  0  0  1158  511   0   0   0   0   0   0   0   111  18   7   0   0  100
```

② 기정출력에 디스크의 전송정보를 추가한다.

```
            vmstat  -d
   procs   memory             page                        faults       cpu
r  b  w  avm   free  re  at  pi  po  fr  de  sr   in  sy  cs  us  sy   id
0  0  0  1158  511   0   0   0   0   0   0   0   111  18   7   0   0  100
Disk  Transfers
             device      xfer/sec
             c0t6d0      0
             c0t1d0      0
             c0t3d0      0
             c0t5d0      0
```

③ 80렬형식에서 기정출력의 표시

```
vmstat -n

VM
memory                    page                           faults
avm   free  re  at  pi  po  fr  de  sr   in   sy   cs
1158  430   0   0   0   0   0   0   0   111  18   7

CPU
        cpu       procs
us    sy     id      r    b    w
0     0     100      0    0    0
```

④ 페지개선 및 주소변환실패를 기정출력에서 교환한 프로세스와 교체한다.

```
vmstat -S
   procs  memory            page                        faults       cpu
r  b  w  avm   free  si  so  pi  po  fr  de  sr   in  sy  cs  us  sy   id
0  0  0  1158  430   0   0   0   0   0   0   0   111  18   7   0   0  100
```

⑤ 5s 구간내에서 기정출력을 두번 표시한다. 머리부분은 반복되지 않는다는것을 지적해 둔다.

```
vmstat 5 2
   procs  memory            page                        faults       cpu
r  b  w  avm   free  re  at  pi  po  fr  de  sr   in  sy  cs  us  sy   id
0  0  0  1158  456   0   0   0   0   0   0   0   111  18   7   0   0  100
0  0  0  1221  436   5   0   5   0   0   0   0   108  65  18   0   1   99
```

⑥ 5s 구간내에서 80렬형식으로 기정출력을 두번 표시한다. 머리부분은 반복되지 않는다는것을 지적해 둔다.

```
vmstat -n 5 2
VM
 memory                   page                           faults
avm   free  re  at  pi  po  fr  de  sr   in   sy   cs
1221  436   0   0   0   0   0   0   0   111  18   7
CPU
           cpu    procs
us    sy     id      r    b    w
```

```
0       0      100        0      0      0
1221   435    2    0    2    0    0    0    0   109      3     17
        0         1        99      0      0       0
```

⑦ 5s 구간내에 80 렬형식으로 기정출력과 디스크전송에 대하여 두번 표시한다. 머리부는 반복하지 않는다는것을 지적해 둔다.

```
vmstat -dn 5 2
VM
 memory                   page                          faults
avm    free   re   at   pi   po   fr   de   sr    in     sy     cs
1221   435    0    0    0    0    0    0    0   111     18      7
CPU
              cpu      procs
us      sy        id          r      b      w
0       0      100        0      0      0
Disk Transfers
 device          xfer/sec
 c0t6d0               0
 c0t1d0               0
 c0t3d0               0
 c0t5d0               0
VM
memory                   page                          faults
avm    free   re   at   pi   po   fr   de   sr    in     sy     cs
1219   425    0    0    0    0    0    0    0   111     54     15
CPU
              cpu      procs
us      sy        id          r      b      w
1       8        92         0      0      0
Disk Transfers
 device          xfer/sec
 c0t6d0               0
 c0t1d0               0
 c0t3d0               0
 c0t5d0               0
```

⑧ 체계시동후에 분기개수와 가상기억페지들의 개수를 표시한다.

```
vmstat  -f
24558  forks,   1471595  pages,   average=  59.92
```

⑨ 페지화관련사건들에 대한 계수기의 값을 표시한다.

vmstat s
0 swap ins
0 swap outs
0 pages swapped in
0 pages swapped out
1344563 total address trans. faults taken
542093 page ins
2185 page outs
602573 pages paged in
4346 pages paged out
482343 reclaims from freee list
504621 total page reclaims
124 intransit blocking page faults
1460755 zero fill pages created
404137 zero fill pages faults
366022 executable fill pages created
71578 executable fill pages faults
0 swap text pages found in free list
162043 inode text pages found in free list
196 revolutions of the clock hand
45732 pages scanned for page out
4859 pages freed by the clock daemon
36680636 cpu context switches
1497746186 device interrupts
1835626 traps
87434493 system calls

경고

출력의 정확한 마당너비와 공간은 체계, HP-UX 공개, 표시될 자료에 많이 의존하여 변한다는것을 vmstat의 사용자는 알아야 한다.

저자

vmstat는 캘리포니아종합대학, 버클리와 HP에 의하여 개발되였다.

파일들

/dev/kmem

관련항목

iostat(1)

제１５장. 공동탁상환경

공동탁상환경은 **탁상수준**(Desktop Level)에서 UNIX 를 통일시키기 위한 주요 UNIX 판매자들의 노력의 결과이다. CDE 는 많은 UNIX 체계상의 X 말단과 **워크스테이션**(Workstation)의 사용자들에 의하여 널리 리용된다. 많은 UNIX 변종들은 CDE 를 수행시키면서 조작할수 있기때문에 여기서는 IBM 의 AIX 체계들, Hewlett-Packard 의 HP−UX 체계들, Sun Microsystems 의 Solaris 체계들에 대한 CDE 를 취급한다. 이 장에서는 CDE 에 대한 개괄을 준다. 즉 CDE 의 형태와 보기방법, CDE 환경의 변경을 서술하며 X, Motif, CDE 들사이의 관계에 대한 일련의 배경을 서술한다. 이 장을 서술하는데 리용된 CDE 방안들은 AIX CDE 1.0, HP−UX CDE 2.1.0 과 Solaris CDE 1.3 이다. 최근의 공개판들에는 일부 특징들이 보충되였는데 일반적으로 그 특징들은 거의 동일한 기능을 수행한다.

CDE 의 개별화를 쉽게 할수 있게 하는 여러가지 특성들이 있다. 매 사용자가 사용하게 되는 **양식관리자**(Style Manager)는 개별적사용자의 요구에 기초하여 CDE 를 쉽게 개별화할수 있게 한다. 그러나 앞으로 가까운 시기에 사용자를 위한 CDE 기능들의 공통적 변경수단도 제공될수 있을것이다. 실례로 만일 많은 사용자들이 수행시킬 응용프로그람이 있다면 환경변수들을 설정할수 있고 **내리펼침차림표**(Pull-Down Menu)들과 편리한 서체 등을 준비할수 있다. 이렇게 사용자들은 더 능률적인 환경을 만들수 있다. 그 다음 사용자들은 파일관리자의 특징에 대한 정의와 배경의 선택과 같은 추가적인 개별화도 수행할수 있다.

이 장에서는 CDE 에 대한 전반적인 리해를 주기 위하여 다음과 같은 문제들을 취급한다.

1) 도형사용자대면부(GUI)는 왜 필요한가？
2) CDE 의 기초개념
3) CDE 의 개별화
4) CDE 의 개선된 문제들
   ㄱ) X, Motif, CDE 들사이의 관계
   ㄴ) X, Motif, CDE **구성파일**(Configuration File)들
   ㄷ) CDE 를 시작할 때 사건들의 렬
   ㄹ) CDE 와 수행

첫째로, 공통적이기는 하지만 다루기 힘든 행별말단대면부보다도 도형대면부를 마련하는 리유를 설명하려고 한다. CDE 탁상화면의 작업공간에 대한 개괄을 두 부분 즉 AIX 와 HP−UX(이것들은 류사하기때문이다.) 그리고 Solaris 로 나누어 주기로 한다. 매 부분에서는 **정면판**(Front Panel)의 특성을 개괄한다. 둘째로, 몇가지 CDE 전용화를 하기 위한 방법을 제시한다. 이 개별화는 자기자신에게 편리한 변경을 기본적으로 할수 있게 한다. 우선 일부 기본적인 변경을 어떻게 하는가를 보여 주고 다음에 새로 체계에 가입할 때와

새 가입통보의 편집과 같은 더 복잡한 변경방법을 보여 준다. 셋째로, X, Motif, CDE 들사이의 관계에서 더 개선된 문제들, 구성파일의 리용과 위치, CDE 를 시작할 때 내부적으로 무엇이 일어 나는가 하는 문제와 CDE 의 수행에서의 일부 문제점들을 고찰한다.

도형사용자대면부(GUI)는 왜 필요한가

탁상화면을 리용하는 매개 콤퓨터들에서 탁상화면은 리용하기 쉽게 만들어 져야 한다. 콤퓨터들을 리용하는 새 방법에서는 지령행재촉문을 피하는 방법이 요구되였는데 이 방법은 사용자들이 복잡한 지령들을 알지 못하거나 망작업과 같은 기술적인 내부구조지식을 가지고 있지 않아도 작업을 할수 있게 하는 방법이다. 지령에 대한 정보가 아직 중요하지 않다고 할수는 없다. 일반콤퓨터사용자들이 사용함에 있어서 이 지령들에 대한 정보는 중요하면서도 너무 전문화되였다고 말할수 있다. 모든 사용자들에게 있어서 응용프로그람들에 대한 지식은 매우 중요한것이였다. 자동차운전수는 기계를 잘 몰라도 운전할수 있는데 콤퓨터사용자는 왜 콤퓨터기술을 알아야 하는가? 도형사용자대면부는 응용프로그람의 **말단사용자**(End-User)들이 콤퓨터를 쉽게 리용할수 있게 한다.

그림 15-1 은 콤퓨터하드웨어, 조작체계, 도형사용자대면부들사이의 관계를 보여 준다. 콤퓨터는 가장 밑바닥에 깔려 있는 하드웨어이다. 다음 웃층에 있는 조작체계는 기호에 기초한 사용자대면부를 표시한다. 이 수준에서 콤퓨터를 조종하려면 사용자들은 건반으로 지령들을 입력하여야 한다. X Windows System 으로 시작되는 다음 3 개의 층들은 도형사용자대면부를 제공한다. 이 수준에서 콤퓨터를 조종하려면 사용자들은 마우스로 도형조종요소들을 조작하여야 한다.

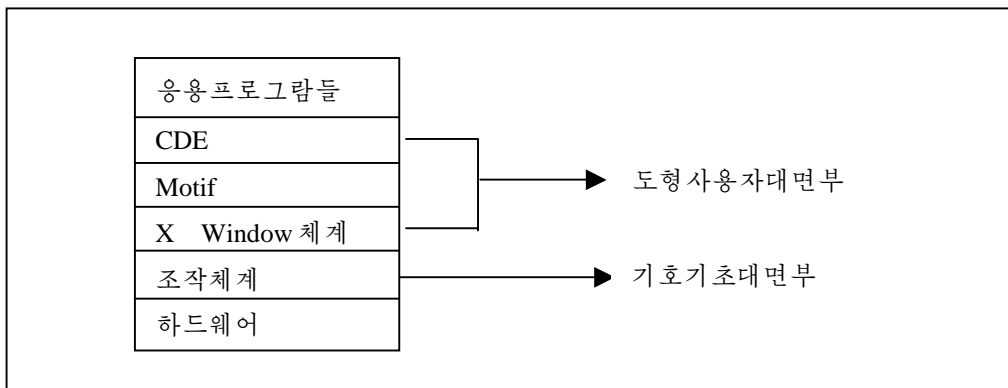



그림 15-1. 사용자대면부성분들

GUI 에서는 지령형식과 인수들을 암기할 필요가 없으며 대신에 창문에서의 탐색을 통하여 필요한 조작들을 수행할수 있다. 사용자는 콤퓨터를 리용하기 위하여 내리펼침차림표, 누름단추, 흐름띠의 밀기와 같은 여러가지 조작을 직접 할수 있다. 어느 한 기능을 수행시키기 위하여 필요한 조작체계의 지령입력은 대폭 감소된다. 콤퓨터는 GUI 를 리용함으로써 배우기 쉽고 리용하기 쉽게 되였다.

CDE는 조작체계를 빨리 동작시키므로 값이 상당히 눅을수 있으나 GUI들은 RAM리용과 그 수행에 관하여 비용이 적게 들지 않는다. GUI들은 그 리용에 드는 비용이 적지 않음에도 불구하고 계산환경의 필수적인 부분으로 되고 있다. 그 보조프로그람들의 리용으로 얻는 리득은 그 비용을 보상할만큼 가치가 있다.

CDE는 숙련을 요구하지 않는 도형조종을 벗어 나서 사용하기 더 쉬운 세계적으로 관심있는 과제들을 제기할수 있게 한다. 이것은 콤퓨터를 리용하는 품을 덜어 준다. GUI는 매 표시장치당 다중창문들을 보장하여 주며 의뢰기-봉사기의 위상적견지에서도 가치가 있는것 등 두가지 리득을 준다. GUI가 제공하는 다중창문의 리용으로 얻는 리득은 매 창문이 개별적응용프로그람을 포함할수 있다는것이다(창문이라는것은 테두리로 둘러 막힌 직4각형령역이다). 사용자는 열린 다중창문과 작업할수 있다. 한 걸음 더 나아가면 CDE는 다중작업공간을 통하여 규정된 작업공간에서 사용자가 과제로서 개별적응용프로그람의 창문을 리용할수 있게 하여 준다. 실례로 이름 Mail을 가진 작업공간에서 사용자는 들어 온 전자우편의 목록을 보기 위한 응용프로그람창문을 가질수 있고 현재 우편통보를 읽을수 있으며 후에 전송을 위하여 통보를 준비할수도 있다. Financials이라는 다른 작업공간에서 매 사용자는 창문에서 여러가지 통보들을 볼수 있다.

의뢰기-봉사기위상은 망을 통하여 전달되는 계산자원들에서 필요한 계산을 할 때 효과적으로 접근하게 된다. 망의 의뢰기-봉사기위상에서 수많은 콤퓨터들은 특정의 목적으로 봉사된다(파일봉사자에서는 파일관리를 하고 응용프로그람봉사자에서는 응용프로그람을 수행한다.). 일부 의뢰기콤퓨터들에서 작업하는 사용자들은 망의 어디서나 원격적으로 파일이나 응용프로그람들에 접근한다. 체계관리자는 파일봉사자를 매개의 개별적의뢰기콤퓨터에로가 아니라 파일봉사자에만 거꿀복사를 하게 하면서 파일거꿀복사의 집중화로 체계관리를 감소시킨다. 이 설정은 규정된 시간구간마다 그 파일들의 거꿀복사를 담보하여 준다. 응용프로그람봉사자는 매 의뢰기콤퓨터에 요구되는 RAM의 크기와 요구되는 디스크의 기억크기와 개수를 감소시키므로 처리비용을 감소시킨다. 응용프로그람의 단일한 방안을 응용프로그람봉사자에 상주시키고 수행하면 그 응용프로그람은 망을 통하여 다중사용자들에 의해 접근될수 있다.

이 위상에 대한 표상은 복잡하게 생각될수있지만 CDE GUI를 쉽게 관리할수 있다. 어떤 파일에 접근하려면 사용자들은 파일관리자의 창문으로부터 한 파일그림기호를 끌어다 놓는다. 응용프로그람을 시작하려면 사용자들은 응용프로그람의 그림기호를 두번 선택한다. 어떤 파일을 인쇄하려면 사용자는 정면판에서 적당한 인쇄기의 그림기호우에 파일을 끌어다 놓는다. 사용자들은 이 파일과 응용프로그람들이 어디 있고 무슨 등록부에 있는지, 무슨 콤퓨터인지, 어떻게 접근되는지 알수 없다. GUI는 배치된 조종과 콤퓨터기계에서의 모호한 내부구조가 아니라 사용자들이 자기의 작업에만 집중할수 있게 하여 준다는데 그 위력이 있다.

공동탁상환경(CDE)의 기초개념

CDE 사용자편람들은 체계의 기능을 소개하므로 여기에서는 CDE 에서 보이는 요소들과 어떤 CDE 를 개별화할 때 작업할 기본령역들의 개괄만을 주기로 한다. 먼저 CDE 를 보는 관점에 따라 류사성과 차이점을 지적하여 준다.

CDE 가입화면은 가입하기전에 여러개의 선택항목을 제시한다. 사용자이름을 입구하는 령역의 아래에는 4 개의 단추들 OK, Start Over, Options, Help 가 있다. 단추 OK 는 가입에 들어 간다음에 암호를 입구하고 Enter 를 누른것과 같다. 단추 Start Over 는 사용자의 가입을 취소하고 다시 시작하게 한다. 단추 Options 는 가입하기전에 만들 필요가 있는 어떤 시동가입기간의 모형을 제공한다. Language 는 기정언어를 변경시킬수 있게 한다. 이것은 다른 언어로 된 류사한 쏘프트웨어를 리용하는데 필요할것이다. 보통 언어가 체계에 미리 적재되였다고 할 때 다른 언어로 넘어 가려면 단순히 CDE 를 선택하는 것으로 리용할 언어를 바꿀수 있다. 가입기간은 CDE 탁상화면의 가입기간 혹은 X 가입기간이지만 CDE 탁상화면이 없는 Failsafe 가입기간으로 가기 위하여 선택항목을 지적하여야 한다. Solaris 는 OpenWindows Desktop 혹은 User' s Last Desktop 에 가입하기 위한 선택항목들을 가진다. Command Line Login 은 CDE 가 없이 기동되는 가입을 하게 한다. 이것은 보통 말단방식의 가입기간으로 된다. Reset Login Screen 은 이름이 무엇인가를 암시하게 한다. 마지막으로 Help 는 이 모든 단추들의 특징을 간단히 서술한다.

가입화면자체는 CDE 가입을 현시하고 AIX 와 같은 조작체계에 가입 혹은 어떤 상사에 가입하는것과 같은것이다. 환경통보는 사용자가 가입에 들어갈 때 나타난다. 사용자이름을 입구한 후에는 암호입구를 예고한다. 일단 암호가 입구되면 정확성은 확인되고 CDE 는 시동되며 탁상화면은 표시된다. Solaris 체계에서 첫번째로 가입할 때 CDE 에 가입하려고 하는가 혹은 OpenWindows Desktop 에 가입하려고 하는가에 대한 선택항목을 제공하여 준다.

CDE 탁상화면은 4 개의 탁상화면작업공간으로 이루어 져 있고 정면판은 이 매개 부분으로 나누어 진다. 정면판은 여러가지 응용프로그람, 지령, 도구들을 리용하여 대면부리용을 쉽게 하도록 한다. 여러개 성분들은 단순히 지적하고 선택하는 방법으로 쉽게 접근할수 있다. 정면판의 성분들은 대상들과 부분판들로 되여있는데 탁상화면작업공간에 접근하게 한다. 어떤 대상들은 선택할 때 다른 대상과 달리 시계와 같은 항목들만 표시하는데 리용된다. 또한 Calendar 혹은 dtmail 과 같은 응용프로그람을 가져 오거나 자물쇠를 잠그거나 나가는 행동의 그림기호와 같은 행동을 수행한다. 부분판들은 접근할수 있는 대상들의 내리펼침차림표이다. 이 대상들은 단순한 항목들을 표시하거나 응용프로그람을 수행시킬수 있다. 기정으로는 AIX 와 HP-UX 에서 Personal Printer 부분판, Personal Applications 부분판, Help 부분판을 볼수 있다. Solaris 에서 모든 판들은 다 부분판들을 포함한다. 부분판들은 다음 항목에서 보게 되지만 CDE 의 개별화(Customizing CDE)와 같이 AIX 와 HP-UX 에 다른판들이 추가될수 있다. 매개 판은 그것을 선택할 때 판우에 작은 화살을 가지기때문에 부분판들을 가진다고 말할수 있다. 정면판의 중심에 4 개 작업공간들이 있다. 이 작업공간은 사용자에게 탁상화면의 번잡함이 없이 개별적작업을

할수 있게 하면서 관련된 과제수행에 필요한 령역을 준비한다.

AIX 와 HP-UX 에서 CDE 는 대단히 류사하지만 Solaris 는 기본 CDE 보기와 느낌에 있어서 지정정면판에 요소들과 부분판들이 더 추가되였다. 여기서는 처음으로 HP-UX 와 AIX 에서 본것처럼 CDE 의 개괄을 주기로 한다. 다음으로 Solaris 의 CDE 에 추가된 기능을 보여 준다.

AIX와 HP-UX에서의 CDE

정면판은 11 개 기본령역 즉 5 개판, 1 개 작업공간령역, 5 개의 추가적 판들로 이루어 진다. 그림 15-2 에서 본것처럼 Clock 로부터 시작하여 Trash can 으로 끝나는 매개 령역을 왼쪽으로부터 오른쪽으로 가면서 개괄한다.

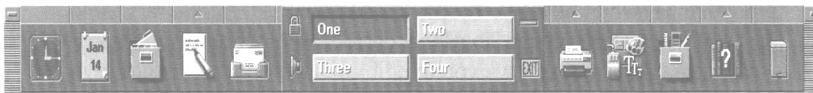



그림 15-2. 정면판

Clock - 현재시간을 표시한다.

Calendar - Calendar 그림기호는 현재 날자를 표시하며 선택될 때 적당한 력서를 표시한다. 력서는 지정과 나머지 설정을 하게 하고 과제목록을 조성하게 한다. 지정은 한번에 한 사건만으로 혹은 순환적으로 설정할수 있다. 한 지정을 소리, 류동통보, e-mail 로 통지할수 있다. 력서와 관련된 지정은 날, 주, 월로 표시될수 있다. 년별력서는 지정없이 표시된다.

File Manager - 파일관리자는 홈등록부와 관련된 파일들을 표시하는 창문을 연다. 여기로부터 파일의 복사, 파일의 이동, 파일의 제거, 프로그람과 스크립트의 수행과 같은 많은 기본적인 파일조작을 할수 있다. 기호적련결의 조성, 파일허락의 변경, 소유관계와 같은 자주 리용되는 연산들은 할수 있다. 파일관리자안에서 파일제거는 영원히 파일을 제거하지 않고 Trash Can 에 파일을 이동시킨다. 이 방법은 만일 파일이 필요될 때 거꿀복사로써 그것을 복구하면서 더 많은 시간을 소비하는것보다 Trash Can 으로부터 그것을 쉽게 복구할수 있다. 그러나 매 가입기간의 끝에서 Trash Can 은 자동적으로 지워진다는것을 알아야 한다. 가입이 취소될 때도 파일들은 영원히 제거된다.

Personal Application - 개별적응용프로그람의 부분판은 CDE 의 본문편집기, dtpad, 말단모방자, dtterm 을 포함한다. 그것은 또한 그림기호편집기를 포함한다. dtpad 는 쉽게 리용할수 있는 완전화면편집기이다. 행별본문편집기인 보통 vi 와는 달리 PC 기호단어처리기와 같이 dtpad 는 본문의 추가, 변경, 제거를 위하여 지시기를 어디에나 이동시킬수 있다. CDE 의 dtterm 은 좋은 기본말단모방자이다. 그림기호편집기는 **비트매프**(Bitmap)(.bm) 혹은 **픽스매프**(Pixmap) (.pm)파일을 열고 그것을 편집할수 있게 한다.

개별응용프로그람의 부분판은 첫번째로 위치하고 있으며 여기서 설정그림기호응용프로그람을 찾을수 있다. 이것은 부분판에 새 그림기호들을 추가할수 있게 한다. 이것은 그림기호가 두번 선택될 때 수행하도록 그림기호와 응용프로그람을 련관시킨다. CDE 전

용화에서 탁상화면을 변경시킬 때 이것을 리용한다.

Mail – 이 그림기호를 선택할 때 CDE 우편통신을 보장하는 dtmail 이 기동된다. 여기로부터 우편통보를 조성할수 있고 통보에 대한 대답, 앞으로 보낼 통보를 구성할수 있다. 대부분의 개선된 전자우편의 특징들은 여기에 다 포함되여 있다. 이 특징에는 통보에 붙이기의 추가와 같은 항목들, 인차 송신자에 혹은 모든 접수자들에 응답보내기, 휴식이라고 알려 주는 통보의 자동적설정과 같은것들이 속한다. 다른 특징은 새 통보가 도착했을 때이며 이때 dtmail 그림기호는 우편함에서 꺼낸 문자를 보여 주도록 변한다.

Workspace Area – 작업공간령역에는 다음 4개 항목들이 있다.

Lock Button – 자물쇠단추는 탁상에서 떨어져 있을동안 자기의 가입기간을 자물쇠로 채울수 있게 한다. 이 안전을 위한 특징은 자기가 없을 때 다른 사람들이 작업상태의 보기 혹은 접근으로부터 보호할수 있게 한다. 이것은 작업에서 리탈할 필요가 있을 때 매번 가입의 탈퇴와 재가입을 피하게 한다. 가입암호나 뿌리암호는 다시 자물쇠를 열 때 입력해야 한다.

Workspace Switch – 4 개의 수들은 기정으로 조성된 개별적작업공간들이다. 이것은 작업공간이 무질서하지 않게 작업을 조직하도록 한다. 작업공간의 리용으로 다른 과제, 응용프로그람 혹은 서로 구별되는 체계들에서 작업을 할수 있게 한다. 작업공간번호 One, Two, Three, Four 의 선택으로 한 작업공간에서 다른 작업공간으로 쉽게 변경할수 있다. 어떤 작업공간에 있는가에는 상관이 없다는것을 지적해 준다. CDE 전용화에서 작업공간에 대한 증가와 이름을 쉽게 변경시킬수 있는 방법을 보게 된다.

Activity Light – 능동성지시기는 아주 단순한데 체계가 작업에 종사할 때 불이 켜진다.

Exit button – 나가기단추는 CDE 가입기간을 종결하면서 CDE 에 가입을 취소한다. 나갈 때 CDE 모형에 따라서 현재 가입기간을 계속하거나 홈가입기간에로 귀환하겠는가를 예고한다. 만일 현재가입기간의 계속을 선택하면 다음에 가입하고 탁상화면은 가능한껏 가입을 취소할 때와 꼭 같게 표시된다. 원격적가입과 같은 일부 경우에는 불가능하지만 응용프로그람을 자동적으로 수행하는것과 같은 경우에서는 가능하다. 만일 홈가입기간에로 귀환을 선택하면 다음에 가입하고 미리 설정한 알려 진 모형으로 귀환될것이다. 이 모형은 간단히 론의된 Style Manager 으로 설정한다.

Personal Printers – 이 부분판은 인쇄관리자(Print Manager)와 같은 지적된 기정인쇄기를 포함하여 체계에서 모형화된 인쇄기들을 포함한다. 정면판의 그림기호는 기정인쇄기의 그림기호이다. 어떤 문서를 인쇄하려면 파일관리자에서 문서그림기호들을 끌어다 인쇄기그림기호우에 놓는다. 인쇄관리자는 순서화된 인쇄파일들을 보기도 하고 그 파일을 인쇄하기전에 순서에서 제거할수도 있다. 그러나 이것은 체계에서 직접 관리되는 인쇄기들과만 작업한다. 오늘의 망으로 된 사무처리에서 인쇄기들은 보통 공유되고 있으나 인쇄기관리자의 기능은 모름지기 앞으로 체계구축에서 봉사기에 있게 될것이다.

Style Manager – 양식관리자는 보는 관점에 따라 많은 시간의 절약 혹은 보기 좋은 탁상화면을 실제적으로 얻을수 있게 하는 방법들중의 하나이다. 여기서는 고유한 선택으로 체계에 가입을 개별화한다. 서체크기변경, 배경, 마우스속도, 피동성으로 규정된 시간이 지난 후 혹은 놀고 있는 규정된 시간이 지난 후에 자동적으로 가입기간을 잠그겠는가 잠그지 않겠는가를 변경시킬수 있다. 이것은 이미 언급한것처럼 매번 가입으로 거꾸

로 가기 위하여 홈가입기간을 설정하거나 체계를 현재가입기간에로 귀환하도록 설정하거나 매번 가입을 취소할 때 질문하는 선택항목을 선택할수 있게 한다. 여기서 취할수 있는 다른 한 모형으로서는 창문초점이 마우스에 뒤따르는가 혹은 작업창문으로 되기전에 창문을 선택해야 하는가를 규정하는것이다. AIX 체계에서는 작업공간들이 정면판에 표시될 때 미리 on(작용) 혹은 off(취소)를 고정할수 있다.

Application Manager – 응용프로그람관리자가 열렸을 때 리용할수 있는 응용프로그람(application)과 행동(action)들을 가진 등록부들을 표시한다. 비록 그것들은 생산자와 판매자들의 조작체계에 따라 다르지만 AIX, HP-UX, Solaris 의 탁상화면응용프로그람들, 탁상화면도구들, 정보와 체계관리 등은 같은 기본특징들을 가진다. 탁상화면응용프로그람들은 Calculator, Man Page Viewer, Icon Editor, Create Action 과 같은 응용프로그람들과 도구들을 포함한다. 탁상화면도구들은 xterm, xwd 포착, 파일압축, 자원들의 재적재와 같은 도구들을 포함한다. 체계관리는 암호변경과 같은 더 일반적인 행동외에 HP-UX 에서는 SAM, AIX 에서는 SMIT, Solaris 에서는 Admintool 과 같은 조작체계마다 특정의 응용프로그람들을 포함한다. 이것을 좀 더 고찰하기로 한다. 개별화된 CDE 정면판에서 통합할것을 바라는 많은 항목들을 찾을수 있을것이다. 여기서 가치를 알수 없는 어떤 자원이 있는가를 Man Page Viewer 로 찾고 실지로 접근할수 있도록 정면판에 그것을 이동시킬것이다.

Help – Help 부분판은 Help Manager, Desktop Introduction, Front Panel 도움말, On_Item Front Panel 도움말와 같은 구조의 결합이다. Help Manager 는 CDE 의 기본적인 직결도움말의 편리한 수단이다. 이 도움말체계는 제목들의 나무와 열쇠를 리용하여 찾아보기능력 혹은 본보기조화를 리용하여 찾기, 도움말관리자에 있는 거꿀추적, 요구한 도움말의 항목의 리력보기 등을 가진 포괄적인 체계이다. Desktop Introduction 은 CDE 의 개괄과 어떻게 작업하는가를 해설한다. Front panel 도움말은 정면판의 그림기호, 부분판, 작업공간의 리용방법에 대한 정보를 준다. On_Item Front Panel 도움말은 요구하는 정면판항목의 선택으로 해설문을 볼수 있게 한다. Help 부분판에서 Help 는 F1 기능건을 눌러서도 요청할수 있다. 만일 체계에 Basic Desktop Customization Help 와 Base Library 이 설치되였다면 AIX 는 그것들을 포함할수 있다.

Trash Can – 파일관리자와 같이 리용되는 Trash Can 은 현재 가입기간에서 제거된 파일들과 등록부들을 보존한다. Trash Can 은 제거되지 말아야 할 파일들을 빨리 복구하게 하는데 리용된다. Trash Can 은 임의의 시간에 영원히 항목을 제거할수 있다. 또한 Trash Can 은 가입기간으로부터 가입을 탈퇴할 때에도 지워 진다.

Solaris에서의 CDE

이미 언급된것처럼 Solaris 는 그림 15-3 에서 보여 주는바와 같이 CDE 에는 많은 부분판들과 부분판들의 항목들이 추가되여 있다. Solaris 가 판과 부분판을 크게 변경시킨 원인은 이미 AIX 와 HP-UX 의 가입기간에서 고찰한것과 기본적으로 같다.

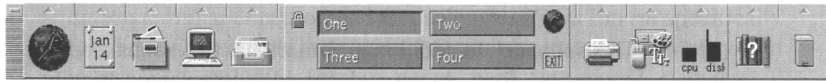



그림 15-3. 정면판

AIX 와 HP-UX 와 같이 Solaris 의 정면판은 11 개 기본령역 즉 5 개 판들, 현재작업공간령역, 5 개의 더 많은 판들로 되여 있다. 여기서는 왼쪽으로부터 오른쪽으로 가면서 World 로 시작하여 Trash Can 으로 끝나는 매 령역을 개괄한다.

Links – 이것은 World 그림기호로 추축할수 있는것처럼 World 에 접근하기 위한것이다. 여기에 Solaris 의 Web browser, HotJava 가 있다. 또한 여기에는 Web browser 의 Personal Bookmarks 에로의 접근과 찾기엔진(search engine)인 Find Web Page 에로의 접근을 포함한 행동 등이 있다. 정면판그림기호에서 지적된것처럼 World 그림기호에 시계를 포함하고 있다.

Cards – 력서는 AIX 와 HP-UX 에서와 같다. 그러나 부분판에 포함된 rolodex 형 주소관리자인 Fine Card 도 있다.

Files – 비록 제목은 다르지만 여기에는 File Manager 가 포함되여 있다. 부분판에 Properties, Encryption, Compress File, Archive, Find File 와 관련된 파일의 행동을 수행할 전문적그림기호들이 있다. Solaris 는 또한 디스크장치와 CD-ROM 을 관리하기 위한 행동들도 포함하고 있다.

Applications – 단순히 재이름화된 Applications 는 Text Editor 뿐아니라 Text Note 와 Voice Note 도 찾을수 있다. 가능한 음성에 대하여 Voice Note 는 WAV, AU, AIFF 형식을 가진 음성파일의 연주, 기록, 보관을 할수 있다. 이 부분판에서는 AIX 와 HP-UX 체계에서 Application Manager 밑에 있었던 Applications 부분판들도 찾아 볼수 있다. 다른 체계에서와 같이 이 부분판은 Desktop Applications, Desktop Tools, Information, System Administration 을 포함한다. 이것들은 기본적으로 모든 체계들에서 같다.

Mail – 이 부분판은 dtmail 과 함께 Suggestion Box 를 더 포함하고 있다. 탁상으로부터 전자우편을 지정된 통보를 리용하여 Sun Microsystems, Inc. 에서 미리 주소화된곳으로 자동적으로 보낸다.

Workspace Area – 작업공간령역에는 다음과 같은 5 개 항목들이 포함된다.

Lock button – AIX 와 HP-UX 에서와 같다.
Workspace switch – AIX 와 HP-UX 에서와 같다.
Progress Indicator – AIX 와 HP-UX 에서 Busy Indicator 와 같다.
Edit button – AIX 와 HP-UX 에서와 같다.
Personal Printers – AIX 와 HP-UX 에서와 같다.

Tools – 여기에 Style Manager 가 위치하고 있으며 이것은 정면판에 표시된다. Tools 는 CDE 오유일지에 쉽게 접근할수 있게 하고 Find Process 체계에서 수행되는 모든 프로쎄스들을 볼수 있게 하거나 선택된 프로쎄스를 없앤다. Cusotmized Workspace Menu 와

Add Item to Menu 행동은 Workspace Menu 를 쉽게 변경시키게 한다. Workspace Menu 는 탁상화면의 빈령역우에서 마우스를 위치화하고 오른쪽 단추를 눌러 접근한다. AIX 와 HP-UX 는 포괄적인 안내를 가지는데 Solaris 는 이 행동에 접근하기 위한 다른 방법으로서 Workspace Manu 에 정면판과 부분판행동을 포함하고 있다.

Hosts – 여기서 Solaris 는 일반적으로 AIX, HP-UX 와 다르다. Hosts 부분판은 체계관련행동들을 포함한다. 정면판에서 우리는 CPU 가 얼마나 바쁘게 작업하며 디스크장치가 얼마나 있는가를 지적하는 Performane Monitor 그림기호를 찾아 볼수 있다. 부분판에서 이 Hosts 는 dtterm 말단모방자를 열고 Console 은 규정된 통보표시를 위하여 dtterm 을 연다. System Information 은 체계이름, 하드웨어모델, 망작업 IP 주소와 그룹, 물리적기억과 가상기억, 조작체계방안, 마지막으로 재시동된 날자와 시간을 포함하는 체계정보를 제공한다. Find Host 는 Cards 부분판에서 Find Card 와 같다.

Help – AIX 와 HP-UX 에서와 같다.

Trash – 이것은 Trash Can 을 비우기 위한 부분판그림기호가 추가된것외에 Trash Can 과 같다.

지금까지 CDE 의 보기와 감각에 대한 개괄을 하였다. 이러한 지식을 가진다면 탁상화면환경에 있는 모든것을 쉽게 볼수 있다. 다음으로 자기의 작업환경에 알맞는 보기와 감각의 변경방법을 고찰한다.

CDE 전용화

임의의 CDE 구성파일들을 변경시키기전에 우선 한가지 수법을 제기한다. 앞에서 언급되였지만 여기서 다시 지적해 둔다.

변경을 시작하기전에 다음과 같은 사항들을 알아야 한다.

1) 사용자들은 무엇이 요구되는가？
2) 이 요구조건들중에 어떤 요구가 CDE 의 재모형화에서 만날수 있는가?
3) 어떤 수준에서 이 변경을 해야 하는가?(체계범위, 사용자그룹, 개별적사용자들만)
4) 변경시켜야 할 CDE 파일들은 어떤것인가?(이름들과 지령들)
5) 무엇을 변경시켜야 하고 파일안에서 그것들의 순서는 어떠한가?

매개 CDE 성분(자원들과 그 값을 보기 위한)들과 매개 CDE 구성파일들을 복사하고 중요한 페지들을 포함시키면서 다루기 쉬운 결합형식을 가지는것이 좋다.

이제 CDE 에서 어떻게 작업하는가에 대하여 보기로 한다. 또한 사용자단체 혹은 매개별적사용자를 위한 체계의 개별화와 변경에 대하여 보기로 한다.

변경시키기전에 변경할 때 무엇이 기동되는가에 대한 리해를 가져야 한다. CDE 를 다방면적으로 보기 위하여 더 개선된 변경과 같은 단순한 변경을 하여야 한다.

임의의 체계에서 이 모든 변경을 할수 있다. 그리고 장치기능과 같은 힘든 기능들을

리용하여 이 과제를 수행하려고 한다면 배경에서 처리되고 있는 기능을 리해하여야 한다. 그리고 변경을 할 때에는 반드시 확인을 하여야 한다.

Style Manager를 리용하여 변경하기

서체크기

CDE에 처음에 가입할 때 작업공간은 그림 15-4에서 보여 준것처럼 표시된다. 많은 사용자들은 우선 서체크기를 변경시킨다. 초기 시동될 때는 4로 설정되는데 대부분 사용자들은 더 큰 서체를 요구하게 된다. 이런 변경은 쉽게 할수 있다.

그림 15-4. Style Manager

1) 정면판에서 Style Manager 그림기호를 누른다. 그러면 Style Manager가 기동된다.
2) Font를 누른다.
3) Highlight 5를 누른다.
4) OK를 누른다.

배경과 색

Style Manager는 **배경막**(Backdrop)과 색들도 변경시킬수 있게 한다.

1) Backdrop를 누르면 배경막을 변경시킬수 있다.
2) 어떤 문양과 같은것을 지적한 후에 apply를 누르고 다음에 배경을 변경시키려면 close를 누른다. 그러나 이 작용은 현재작업공간에서만 배경막을 변경시킨다는것을 알아야 한다.

다른 작업공간들을 변경시키려면 그 작업공간에 가서 Style Manager를 기동시키거나 이미 Style Manager가 기동되여 있으면 창문에서 File 우의 오른쪽 구석에 있는 작은 "-"을 눌러 내리펼침차림표에 접근할수 있다. 여기서 Occupy All Workspaces를 선택할수 있다. 그러면 다른 작업공간에 쉽게 갈수 있으며 Style Manager는 그곳에서도 리용될수 있다. 배경구역은 다른 작업구역에 이동될 때 고려된다.

3) 색을 변경시키려면 Color 그림기호를 누른다. 그러면 선택할수 있는 여러가지 색들의 목록이 나타난다. 만일 목록에 요구되는 색이 없다면 쉽게 색,

색채, 광도, 대조를 변경시킬수 있다. 인터네트의 영상과 같은 다른곳에서 색을 취하여 색도식에 포함시킬수 있다. 일단 요구하는 색을 만들었다면 고유한 이름으로 만든 색들을 보관시킬수 있다.

정면판으로부터 대상의 추가 혹은 제거

정면판에 흔히 리용되는 항목을 포함시키면 더 쉽게 작업할수 있다. 많은 경우에 자주 리용하는 작용들중의 하나는 말단창문을 여는것이다. 비록 CDE 가 기정말단창문으로서 dtterm 을 가지지만 때때로 xtern 창문을 리용하려고도 한다. dtterm 이 AIX 와 HP-UX 체계에 존재하는 Personal Application 부분판에 xtern 을 추가한다. Solaris 사용자들은 호스트말단모방자가 있는 호스트부분판에 xterm 을 놓을수 있다.

CDE 정면판에 대상을 추가하는 방법에는 다음과 같은 두가지가 있다.

- 부분판에 그 대상을 끌어다 놓고 그것을 기정부분판으로 한다.
- /etc/dt/appconfig/types/c/dtwm.fp 구성파일을 변경시킨다. 이 방법은 행동을 조성한 후에 파일행동을 조성할 때 리용된다.

끌어다 놓기로 조종단추를 추가할 때 기본단계들은 다음과 같다.

- 응용프로그람관리자의 보기로부터 요구되는 응용프로그람그림기호를 정면판단추로 끌어다가 부분판의 설정부분(꼭대기부분)에 놓는다.
- 그림기호우에 마우스지시자를 놓고 부분판안내를 표시하기 위하여 마우스단추 3 을 누른다.
- Copy to Main Panel 을 누른다.

1) AIX 나 HP-UX 에서 혹은 Solaris 에서 호스트부분판의 Personal Applications 부분판에서 웃방향화살건을 누른다.
2) Desktop Applications 와 Desktop Tools 가 있는 Application Manager 그림기호를 누른다.
3) Desktop-Tools 를 두번 누른다. 여기서 Xterm 을 찾는다.
4) HP-UX 와 AIX 의 Personal Applications 부분판 혹은 Solaris 에서 호스트꼭대기에 있는 Install Icon 창에다 Desktop-Tools 로부터 xterm 그림기호를 끌어다 놓는다.

이제 만일 정면판에 xterm 을 포함시키려고 한다면 다음과 같이 할수 있다.

1) Personal Applications 혹은 호스트부분판에서 xterm 그림기호를 오른쪽 화살건으로 누른다.
2) 리용한 CDE 에 의존하여 Copy to Main Panel 혹은 Promote to Front Panel 을

선택한다.

다른 작업공간의 추가

정면판을 변경하는 다른 하나의 쉬운 방법은 다른 작업공간을 추가하는것이다. CDE에는 기정으로 4개 작업공간이 있지만 이것은 쉽게 변경될수 있다. 이제 Web View라고 부르는 1개 작업공간을 더 추가하는 방법을 보기로 한다. 이 작업공간은 인터네트에 접근할 창문으로 리용될수 있다.

1) 정면판의 작업공간령역에 마우스를 배치하고 마우스의 오른쪽 단추를 누른다.
2) 내리펼침차림표로부터 Add Workspace를 선택하면 작업공간 New가 추가된다.
3) New로 표시된 새 작업공간을 마우스의 오른쪽 단추로 선택하고 Rename을 누른다. Web View를 입구하고 Return을 누른다.

만일 어떤 작업공간을 제거하려면 삭제할 작업공간을 마우스의 오른쪽 단추로 선택하고 다음에 delete를 누른다. 이러한 변경은 쉽고 이것은 개별적사용자의 작업공간을 개별화하는데 도움을 준다.

다른 방법으로 정면판의 변경

단추들의 추가와 제거외에 다른 방법으로 정면판의 형태를 변경시킬수 있다. 이 다른 방법들은 기정값을 변경시키기 위하여 다음과 같은 Workspace Manager의 자원들을 리용한다.

- clintTimeOutInterval - 바쁘게 작업이 진행될 때의 불켜기시간의 길이이다. 지적자는 의뢰기가 정면판에서 시작될 때 물시계로 표시된다.
- geometry - 정면판의 x와 y자리표위치이다.
- highResFontList - 높은 분해도표시로 리용될 서체이다.
- lowResFontList - 낮은 분해도표시로 리용될 서체이다.
- mediumResFontList - 중간 분해도표시로 리용될 서체이다.
- name - 다중정면판이 dtwm.fp에 있을 때 리용될 정면판의 이름이다.
- pushButtonClickTime -(우연히 응용프로그람을 두번 시동시키는것을 피하기 위하여) 마우스의 두번 누르기와 두개의 개별적선택을 구별할 시간구간이다.
- waitingBlinkRate - 정면판의 Busy지시기의 깜박거리기비률이다.
- workspaceList - 작업공간이름의 목록이다.
- title - 작업공간단추에 나타날 제목이다.

다른 모든 작업공간관리자의 자원들과 마찬가지로 이 정면판자원들은 다음과 같은 형식을 가진다.

```
Dtwm * Screen * resource: value
```

실례로 정면판은 기정으로 4 개의 작업공간파일대신에 이름들이 각각 Mail, Reports, Travel, Financials, Projects, Studio 인 6 개 작업공간을 포함시킨다. 이 작업공간들은 New Century Schoolbook, 10 개 점, 강조체로 규정된 큰 서체를 리용한다. 이때 체계관리자로서 다음과 같이 자원을 규정하면 누구나 다 만족할수 있다.

```
Dtwm*0*workspaceList: One Two Three Four Five Six
Dtwm*0*One*title: Mail
Dtwm*0*Two*title: Reportes
Dtwm*0*Three*title:          Travel
Dtwm*0*Four*title: Financials
Dtwm*0*Five*title: Projects
Dtwm*0*Six*title:            Studio
Dtwm*0*highResFontList:
                             -adobe-new century scrollbook-bold-r-normal\
                             - -10-100-75-75-p-66-iso8859-1
```

영상과 겹친 판이 모두 가능한 표시를 제외하고 화면지정은 보통 0 이다. X*screens 파일에서 화면의 순서는 화면번호로 결정된다. 첫 화면은 보통 영상판이고 0 으로 지정되며 자원규정에서 6 개 제목의 작업공간이름(One, Two, Three, Four, Five, Six)을 포함한다는것을 보여 준다.

이 장의 뒤부분들에서 상세히 론의되는 우와 같은 변경은 sys.resources 파일에 첨가된다. 이 변경은 매 사용자의 RESOURCE_MANAGER 성질에 새 자원행들을 삽입하기 위하여 EditResources 행동을 리용할수도 있고 작업공간관리자를 재시동할수도 있다.

여기서 부족점은 물리적으로 매 사용자의 작업령역으로 가서 몇분동안씩 조작을 해야 한다는것이다. 우점은 변경이 직접 진행되며 현재가입기간을 복구하려는 사용자들에 대하여 정확히 dt.resources 에 자동적으로 보관된다는것이다. 만일 사용자가 가입하고 있을동안 dt.resouces 파일을 덧쓰기로 변경한다면 오유가 나타날수 있다.

Slide-Up부분판들에서의 변경작업

정면판과 정면판조종을 정의한다음 부분판들은 dtwm.fp 에서 정의된다. 정면판조종단추와 부분판의 련관을 설정하기 위하여 정면판조종이름은 부분판정의에서 용기의 이름으로 목록화된다.

주의 : /etc/dt 는 **전역적변경**(Global Change)이 만들어 진 위치이다. $HOME/.dt 는 국부적 혹은 개별적사용자의 변경이 진행되는 위치이다.

정면판에 slide-up 부분판을 첨가하기 위하여 다음과 같은 걸음들을 준다.

1) /usr/dt/applconfig/types/C/dtwm.fp 파일을 /etc/dt/appconfig/types/C/dtwm.fp 혹은 $HOME/.dt/types/dtwm.fp에 복사한다.
2) silde-up과 관련된 조종단추들을 결정한다.
3) dtwm.fp 에서 부분판의 정의파일을 조성한다. 이것은 다음과 같은 형식을 취한다.

```
SUBPANEL  SubPanelName
{
CONTAINER_NAME  AssociatedForntPanelControlButton
TITLE  SubPanelTitle
}
```

4) 부분판에 대한 조종의 정의를 조성한다. 이것은 다음과 같은 형식으로 한다.

```
CONTROL  ControlName
{
TYPE  icon
CONTAINER_NAME  SubPanelName
CONTAINER_TYPE  SUBPANEL
ICON  BitmapName
PUSH_ACTION  ActionName
}
```

정면판조종단추에서와 마찬가지로 처음부터 시작하는것보다 이미 존재하는 부분판파일을 복사하고 그것을 변경시키는것이 더 쉬울수 있다.

기정인쇄기에 대한 이름표시의 변경

이제 slide-up 부분판들중의 한개 판에서 변경을 쉽게 할수 있다. 만일 Personal Printers 부분판이 나타나 있다면 모형화된 지정인쇄기를 볼수 있지만 그 이름은 보이지 않는다. 정면판파일인 dtwm.fp 에로 거꾸로 가서 다음과 같은 순서로 변경한다.

1) dtpad 혹은 vi 와 같은 마음에 드는 편집기를 앞으로 가져오고 $HOME/.dt/types/dtwm.fp 를 편집한다.
2) 인쇄기를 조종하기 위하여 통보를 아래로 흘러 보낸다. LABLE 이 기정이라는것을 볼수 있을것이다. 기정인쇄기의 이름을 그 행에 추가하거나 변경시

킨다. 실례로 인쇄기를 a464로 할수 있다.

LABLE Default - a464

3) 파일을 보관하고 작업공간관리자를 재시동한다. 작업공간의 빈 령역우에 마우스를 위치화하고 마우스의 오른쪽 단추를 누른다. Restart Workspace Manager를 선택한다. Solaris에서 개별화된 작업공간안내를 가져 오기 위하여 마우스의 오른쪽 단추를 누른다. 여기로부터 Restart Workspace Manager 건을 찾을수 있는 Windows를 선택한다.

기정인쇄기가 실지로 있는가를 보려면 Personal Printers 부분판을 꺼내면 된다. 만일 기정인쇄기를 변경하려면 다시 정면판을 변경시켜야 한다. 그러나 그것을 어떻게 하는가에 대하여서는 아직 모르고 있다. 그림 15-5는 정면판을 보여 준다.

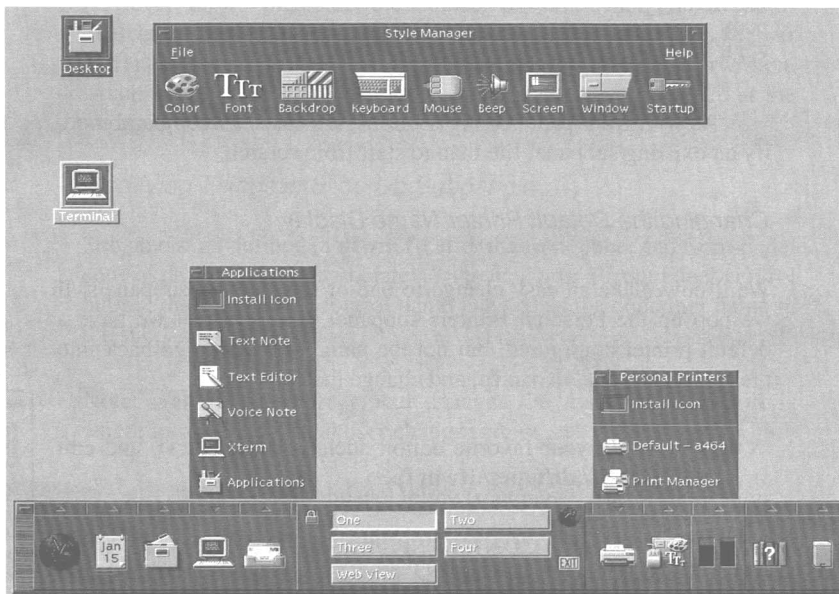



그림 15-5. 변경된 정면판

정면판동화상

정면판 혹은 slide-up부분판에서 작은 지대의 동화상은 비트매프의 지속적인 렬을 표시하는것으로 나타난다. 보통 비트매프들은 /usr/dt /appconfig/icons에 있다. 동화상을 정의하는 비트매프의 목록은 dtwm.fp의끝에 포함된다.

작은 지대의 동화상렬을 조성하자면 다음과 같이 하면 된다.

1) 비트매프들의 지속적인 렬을 조성한다.

2) 다음 형식을 리용하여 적당한 구성파일에 이 비트매프파일들의 한 목록을

추가한다.

```
ANIMATION  AnimationName
{
 bitmap0
 bitmap1
 bitmap2
 bitmap3
 bitmap4
 bitmap5
}
```

3) 다음 형식을 리용하여 적당한 조종정의로 한개 행을 추가한다.

```
DROP_ANIMATION  AnimationName
```

작업공간안내에 항목들의 추가

작업공간의 안내는 sys.dtwmrc 파일에서 정의된다. Solaris 의 정면판에 대한 개팔에서 언급한것처럼 작업공간안내는 탁상화면의 빈령역우에 마우스를 위치화하고 마우스의 오른쪽 단추의 누르기로 표시된다. AIX 와 HP-UX 에서 일반안내가 어디에서 나타나는가? Solais 는 이 행동에 접근을 위한 다른 방법으로서 작업공간안내에서 정면판과 부분판행들을 혼합하였다. 작업공간안내를 리용하는데서 마우스의 왼쪽 단추를 리용하여 모든 세개 체계들을 위한 개별적인 작업공간안내를 조성할수 있다.

개별화된 작업공간안내에는 많이 리용될 지령들과 응용프로그람들이 포함될수 있다. 개별화된 안내는 일반적으로 안내를 보고 선택하는데 마우스의 왼쪽 단추를 리용한다. 작업공간안내는 규칙적으로 진행할 작업의 모임으로서 리용을 편리하게 하여 준다. 실례로 C++을 리용하는 프로그람작성자는 항목 vi 와 isql 들을 이 안내에 포함시킬수 있거나 혹은 체계관리자가 df, ps -ef, netstat 로 안내를 포함시킬수 있으며 망에서 안내들은 체계가입의 한개 안내로 할수도 있다.

한개 혹은 두개의 변경을 위하여 이미 존재하는 작업공간안내를 변경시킬수 있다. 주요한 변경을 위하여 sys.dtwmrc 에서 완전히 새로운 안내정의를 삽입하는것이 더 쉽다는것을 알수 있다.

안내정의는 다음과 같은 형식을 가진다.

```
Menu  MenuName
{
 "Menu  Name"      f.title
 "Frame"           f.exec  /nfs/system1/usr/frame/bin/maker
```

```
 "Second Item"     action
 "Third Item"      action
}
```

첫번째 행은 Manu에 뒤따르는 안내이름을 규정한다. 대괄호들사이에 있는 행들은 표시될 순서로 안내에 나타날 항목들의 목록이다. 이리하여 첫번째 행은 안내이름인데 이것은 f.title로 지정된다. 두번째 행은 분배된 환경에서 원격응용프로그람봉사기로부터 FrameMaker를 시작하도록 정의하기 위한 한가지 실례이다. 대략 45개의 다른 기능들도 있다. 완전한 목록을 보려면 dtwmrc(4)의 기본페지를 참고하면 된다.

안내를 표시할 사용자들에 대하여 f.menu MenuName 행동을 리용하여 마우스단추와 안내정의 그리고 화면위치를 묶는것이 필요하다. 실례로 만일 지시자가 작업공간배경에 있을 때 사용자들이 마우스의 세번째 단추를 눌러 안내를 표시하려면 sys.dtwmrc의 끝에서 Mouse Button Bindings Description부분에 다음과 같은 행을 삽입하여야 한다.

```
<Btn3Down> root f .menu MenuName
```

(실제로 둘째 행에서 DtRootMenu를 위한 MenuName을 변경시키는것은 이미 있던 행을 변경시키는것보다 쉬울것이다.)

vi 편집기를 수행하면서 FrameMaker의 수행을 포함시키는것과 원격체계에 가입하는 것과 같은 안내를 조성하기로 한다.

1) 처음으로 /usr/dt/config/C/sys.dtwmrc 파일을 /etc/dt/config/C에 복사하는것이 필요하다. 즉

```
cp/usr/dt/config/C/sys.dtwmrc /etc/config/C/sys.dtwmrc
```

만일 어느 한 사용자만을 위하여 국부적변경을 하려고 한다면 파일은 /$HOME/.dt에 복사되여야 하고 dtwmrc라는 이름을 변경시켜야 한다. 사용자 admin1에 대하여 그 이름은 /home/admin1/.dt/dtwmrc 혹은 Solaris에서 /home/admin1/.dt/C/dtwmrc에 추가되여야 한다.

2) 리용에 편리한 편집기에서 파일을 변경시킨다. 등록된 DtRootMenu 항목뒤에 새 안내를 추가한다. 체계에서 작업공간안내를 마우스단추들을 리용하여 볼수 있다. 이것을 위해서는 다음과 같이 추가해 주어야 한다.

```
Menu AdminMenu
{
 "Admin' s Menu" f.title
 "Frame Maker"          f.exec "/nfs/system1/usr/frame/bin/maker"
 "VI Editor"            f.exec "xterm -e/usr/bin/vi"
```

```
"Login systemA" f.exec "xtern -geometry 80x50+830+0 -sl 200 -bg
DarkOrchid4 -fg white -n SYSTEMA -T SYSTEMA -e remsh systemA &"
}
```

f.title 의 기능은 f.title 이 menu 제목이라는것을 의미한다. f.exe 는 다음 행을 수행하여야 한다는것을 의미한다. vi 와 가입은 둘 다 수행할 때 말단창문을 요구하고 FrameMaker 는 소속되여 리용되므로 말단창문이 요구되지 않는다. 또한 말단기억과 규정된 색을 리용하여 큰 말단창문을 취하면서 가입을 위하여 xterm 을 표시한다. 그러면 작업공간관리자를 재시동하고 새로운 안내를 표시하게 된다.

3) 작업공간의 빈 령역우에 마우스를 위치화하고 마우스의 왼쪽 단추를 누른다. 그리고 Restart Workspace Manager 를 선택한다.

내리펼침차림표조성도 안내에 부분안내추가도 쉽게 확장될수 있다. Menu 이름뒤에는 f.menu 가 놓인다. "Adminl's Menu"뒤에 다음과 같은것을 추가한다.

```
"Work Menu"        f.menu WorkMenu"
```

다음에 Menu AdminMenu 부분의 } 뒤에 다음과 같이 추가한다.

```
Menu WorkMenu
{
.
}
```

이 개선된 기능을 리용하는것은 실지로 힘든것이 아니다. 가장 힘 든 부분은 전역적변경을 위하여 /etc/dt 에 혹은 개별적변경을 위하여 $HOME/.dt 에 파일을 출구할 등록부들을 기억시키는것이다.

조종단추, 행동, 파일형의 조성

행동은 쉘스크립트 혹은 응용프로그람과 같은 프로쎄스를 시작한다는것을 의미한다. 행동은 단추를 누를 때 시작될 정면판단추와 련결될수 있다. 행동은 자료파일이 작은 지대에 놓일 때 정면판의 작은 지대와 련결될수 있다. 행동은 그림기호의 두번 선택으로 시작될수 있는 파일관리자창문에서 어느 한 그림기호와 련결될수 있다. 행동은 자료파일의 그림기호를 두번 선택하여 행동을 시작하고 자료파일을 열도록 특이한 자료형과 련결시킬수 있다.

사용자의 요구를 만족시키도록 정면판과 기정가입기간을 설정하면서 행동과 자료형들을 조성하는것보다 계산파일을 리용하는것이 더 쉽다.

CDE 행동과 자료형들은 .dt 의 끝에 있는 파일들에서 정의된다. 대부분 다른 CDE 구성파일들과 류사하게 *.dt 파일들은 사용자의 개별적등록부에 복사되고 개별적으로 개성화될

수 있는 체계범위방안을 가진다. 대부분 체계범위의 *.dt 파일들은 /usr/dt/appconfig/types/C 에 있다. 개인의 *.dt 파일들은 /usr/dt/appconfig/types/C 로부터 $HOME/.dt/types 에 복사하여 조성한다. CDE 가 행동을 보기 위하여 리용하는 기정찾기경로와 파일형들은 목록화된 순서로 다음과 같은 많은 등록부들에 포함된다.

- $HOME/.dt/types
- /etc/dt/appconfig/types
- /usr/dt/appconfig/types

DTDATABASESEARCHPATH 환경변수를 리용하여 찾을 경로에 대한 여러개 등록부를 추가할수 있다. 이 환경변수를 삽입하고 체계범위의 작용을 위하여 새로운 찾기경로들을 /etc/dt/config/Xsession 으로 설정한다. 환경변수를 삽입하고 개별적사용자에 위하여 찾기경로를 $HOME/.dtprofile 로 설정한다.

Control Buttons － 조종단추들의 기본조종에 대한 정의는 6 개 부분으로 되여 있다.

- CONTROL name － 정의이름이다. 대괄호밖에는 정의부분만이 있다.
- TYPE － 조종의 형이다. 여기에는 여러개 형들이 있다. 공백과 그림기호는 정면판을 개별화하는데 유용하다. 공백은 공간을 남겨 놓는데 리용된다. 그림기호는 행동 혹은 응용프로그람을 시동시키거나 작은 지대에 놓일수 있다.
- ICON － 정면판에 표시될 비트매프이다. 정면판의 비트매프는 /usr/dt/appconfig/icons 등록부에 들어 있다.
- CONTAINER_NAME － 조종을 유지하는 **용기**(Container)의 이름이다. 이것은 dtwm.fp 에서 목록화된 실지 용기의 이름과 대응되여야 한다.
- CONTAINER_TYPE － 조종을 유지하는 용기의 형을 규정한다. 이것은 BOX, SWITCH, SUBPANEL 로 될수 있으나 그것은 용기이름의 형과 일치되여야 한다.
- PUSH_ACTION － 이것은 조종단추가 눌러 질 때 무엇이 나타나는가를 규정한다.

PUSH_ACTION 은 여러개 가능한 행동들중의 하나이다. 더 많은 정보를 보려면 dtwm 기본페지를 참고하면 된다.

정면판에서 조종단추를 제거하려면 CONTROL 정의행의 제일 왼쪽 란에서 기호 ＃를 입구하여야 한다. ＃는 조종규정을 설명행으로 전환시킨다.

Dtwm.fp 파일의 편집으로 조종단추를 추가하려면 다음과 같이 한다.

1) /usr/dt/appconfig/types/C로부터 dtwm.fp를 /etc/dt/appconfig/types/C에 복사한다.
2) 다음의 형식으로 새 조종의 정의를 추가한다.

```
CONTROL  NewControl
{
```

```
TYPE                    icon
CONTAINER_NAME          Top
CONTAINER_TYPE BOX
ICON                    NewControlBitmap
PUSH_ACTION             NewControlExecutable
}
```

행동과 파일형들은 쌍으로 되여 있으며 자기의 고유한 등록부위치에 있어야 한다. 이 쌍들은 부분판에서 어떤 행동을 조성할 때 지원된다. 행동 혹은 파일형정의를 조성할 위치들은 다음과 같이 하여야 한다.

- /etc/dt/appconfig/types 등록부에 완전히 새 파일을 조성한다. 이 파일은 체계범위영향을 가진다. 파일은 .dt 확장자로 끝나야 한다는것을 기억하여야 한다.
- /usr/dt/appconfig/types 로부터 /etc/dt/appconfig/types 등록부에 user-prefs.dt 를 복사하고 체계범위리용을 위한 정의를 삽입한다.
- $HOME/.dt/types 에 user-prefs.vf 를 복사하고 개별적사용자를 위한 정의를 삽입한다.

대표적인 행동은 다음과 같은 형식을 가진다.

```
ACTION    ActionName
{
TYPE      type
keyword   value
keyword   value
}
```

실례로 FrameMaker 행동은 다음과 같다.

```
ACTION    FRAME
{
TYPE      COMMAND
WINDOW-TYPE  NO-STDIO
EXEC-STRING  /nfs/hpcvxmk6/usr/frame/bin/maker
}
```

대표적인 자료형은 다음과 같은 형식을 가진다.

```
DATA_ATTRIBUTES              AttributesName
{
```

```
    keyword          value
    keyword          value
    ACTIONS          action,  action
}
DATA_CRITERIA
{
    DATA_ATTRIBUTES  AttributesName
    keyword          value
    keyword          value
}
```

모든 정의들은 다음과 같은 일반형식을 가진다.

```
KEYWORD value
```

자료형의 정의는 실지로 두 부분 즉 속성부분과 기준부분으로 되여 있다. 속성부분은 자료형의 보기를 규정하며 기준부분은 행동을 규정한다.

아래의 실례에서는 행동 FRAME 을 리용하는 파일 FrameMaker 들에 대한 파일형을 보여 준다.

```
DATA_ATTRIBUTES          FRAME_Docs
{
    DESCRIPTION          This file type is for FrameMaker documents.
    ICON                 makerIcon
    ACTIONS              FRAME
}
DATA_CRITERIA
{
DATA_ATTRIBUTES_NAME        FRAME_Docs
NAME_PATTERN             *.fm
MODE      f
}
```

이 형식을 리용하여 출발점으로부터 행동과 파일형을 조성할수 있다. 그러나 CreateAction 도구를 리용하는것은 행동을 조성하는 방법보다 더 쉽다. CreateAction 은 Applications Manager 의 Desktop Applications 등록부에 있으며 행동정의를 포함하는 action.dt 파일의 조성을 총체로 지시하는 완전히 빈 대화칸을 제공한다. 다음에 행동하게 하는 영향범위의 적당한 등록부에 이 파일을 이동시킬수 있다. 즉 체계범위영향인 경우 /etc/dt/appconfig/types, 개별사용자인 경우에 $HOME/.dt/types 를 리용할수 있다.

새 그림기호와 행동조성

그림기호를 조성하려면 설명을 좀 더 하여야 한다. 그림기호를 조성하려면 Desktop_Apps 에서 Icon Editor 를 찾아 리용할수 있거나 같은 그림을 찾아 리용할수 있다. 정면판에서 정확히 보이도록 하기 위하여 그림을 끌어다 놓을 때 주의해야 할것은 32*32 화소보다 크면 안된다는것이고 혹은 그림기호의 부분만이 표시되면 안된다는것이다. Icon Editor 에서 그림을 보면 그림의 크기를 알수 있다. 유용한 그림기호를 응용프로그람의 등록부에서 찾을수 있거나 인터네트에서 끌어다 놓을수 있다. 이에 대한 실례로 HP-UX 에 있는 Instant Information 쏘프트웨어를 리용한다. 이것은 CD 에서 응용프로그람들의 모임이다. 경로는 리용될 쏘프트웨어와 대응되여야 한다. 또한 거짓사용자홈등록부인 /home/admin1 을 리용한다.

1) Desktop_Apps(HP-UX), Desktoptools(AIX) 혹은 Solaris 에서 Desktop_Tools 로부터 Icon Editor 를 가져 오고 Application Manager 에서 모든것을 처리할수 있다.
2) File -> Open 을 선택한다.
3) 경로 혹은 등록부이름 /opt/dynatext/data/bitmaps 을 입력한다.
4) 파일이름 logoicon.bim 을 입력한다.
5) Open 을 선택한다.

그림기호를 보고 실지로 32 × 32 화소인가를 검사한다. 이제 .dt 등록부에 그림기호를 보관하여야 한다.

6) File -> Save As 를 선택한다.
7) 경로 혹은 등록부이름 /$HOME/.dt/icons 을 입력한다. admin1 사용자에 대하여 /home/admin1/.dt/icons 로 되여야 한다.
   만일 이 과제를 전역적으로 수행하려고 한다면 /etc/dt/appconfig/types/C 에 이것을 넣어야 한다.
8) 파일이름을 입력하거나 체계에 규정된 그대로 취한다.
9) Save 를 선택한다.

이제 그림기호를 가진다고 할 때 행동파일과 서술파일을 조성한다.

1) 편리한 Text Editor 를 리용하여 다음과 같이 입력한다.

```
ACTION instinfo
{
LABEL           instinfo
TYPE            COMMAND
WINDOW_TYPE     NO_STDIO
```

```
EXEC_STRING          /opt/dynatext/bin/dynatext
DESCRIPTIONThis  action  starts  Instant  Information
}
```

LABEL 은 행동의 이름이고 TYPE 는 지령이며 WINDOW_TYPE 는 자기의 창문을 가지므로 규정하지 않는다. (표준 I/O 가 아니거나 NO_STDIO 이다.) EXEC_STRING 은 수행할 지령이고 DESCRIPTION 은 행동이 무엇을 하는가에 대한 스크립트이다. NO_STDIO 는 련결선이 아니라 밀선을 가진다는것을 알아야 하며 마지막에 기호 "}"를 반드시 써야 한다. 이 두가지를 확인한다음 왜 행동이 나타나지 않으며 만일 나타나면 왜 작업하지 않는가를 료해하는 실험을 진행할수 있다.

2) /$HOME/.dt/types/instinfo.dt 로 이 파일을 보관한다. admin1 사용자인 경우 /home/admin1/.dt/types/instinfo.dt 로 된다. 만일 이것이 전역적모형이라면 파일을 /etc/dt/appconfig/types/C/instinfo.dt 로 보관하여야 한다. 이로부터 어느 한 행동이 만들어 진다. 다음으로 서술파일을 조성한다. 이 파일은 두개 행을 가지는 새 파일이다. 이 파일의 내용은 적합하지 않으나 허락은 수행될수 있는것으로 되여야 한다.

3) 다시 편리한 편집기를 리용하여 입력한다.

```
ACTION  instinfo
DESCRIPTION  This  action  starts  Instant  Information
```

4) 이 파일을 /$HOME/.dt/appmanager/instinfo 로 보관한다. admin1 사용자인 경우 /home/admin1/.dt/appmanager/instinfo 이고 만일 이것이 전역적모형이라면 파일을 /etc/dt/appconfig/appmanager/instinfo 로 보관하여야 한다.

5) 허락을 수행할수 있도록 다음과 같이 변경한다.

```
chmod   555  /$HOME/.dt/appmanager/instinfo
```

우에서 조성한 그림기호와 행동을 포함시키기 위해서는 정면판을 변경시켜야 한다. 정면판파일인 dtwm.fp 는 등록부 /usr/dt/appconfig/types/C 에 들어 있다. 체계파일에 덧쓰기를 요구하지 않으므로 파일을 국부적으로 복사하고 변경시킨다.

1) AIX 와 HP-UX 에서는 /$HOME/.dt/types/dtwm.fp 에, Solaris 에서는 /$HOME/.dt /types/fp_dynamic/dtwm.fp 에 이 체계파일을 복사한다. admin1 사용자인 경우 AIX 와 HP-UX 에서는

```
cp/usr/dt/appconfig/types/C/dtwm.fp  /home/admin1/.dt/types/dtwm.fp
```

이고 Solaris 에서는

```
cp/usr/dt/appconfig/types/dtwm.fp  /home/admin1/.dt/types/fp-dynamic/dtwm.fp
```

이다.

2) 편리한 편집기를 리용하여 dtwm.fp 를 변경시킨다. 그 파일에서 본것처럼 정면판이 표시된것과 같은 순서로 변경시킨다. 시계는 파일에서의 첫 CONTROL 이고 정면판에서의 첫 항목이다. 또한 POSITION_HINTS 는 1 이다. 그뒤에 년, 월, 일이 있고 POSITION _HINTS 는 2 이다. 그다음 POSITION_HINTS 는 4 이며 Text Editor CONTROL 뒤에 새 그림기호와 행동을 기입한다.

3) CONTROL Mail 전에 CONTROL TextEditor 의 끝에 있는 기호 "}"로 간다. 이 위치에서 아래애서 보여 준것과 꼭 같은것을 입력한다. 대문자와 소문자는 정확히 그대로 입력되여야 한다.

```
CONTROL  Info
{
        TYPE                    icon
        CONTAINER_NAME          Top
        CONTAINER_TYPE          BOX
        POSITION_HINTS          5
        ICON                    logoicon.bm
        LABEL                   Instant  Info
        PUSH_ACTIONinstinfo
}
```

4) 다음 항목에서 POSITION_HINTS 는 재번호화되여야 한다. 그러나 Mail 로 시작하여 Trash 로 끝나는 다음의 8 개 항목만 재번호화되여야 한다.

5) 파일을 보관하고 Workspace Manager 를 재시동한다. 만일 모두 정확히 리용하였다면 새로운 그림기호가 나타날것이며 이 그림기호를 선택할 때 그림 15-6 에서 보여 준 Instant Information 이 표시된다.

6) 작업공간에서 빈 령역에 마우스를 위치화하고 마우스의 오른쪽 단추를 누른다. 그다음에 Restart Workspace Manager 를 선택한다.

행동을 조성할 때 기억해야 할 두가지 문제가 있다. 첫째로 PU-SH_ACTION 와 파일이름들이 같아야 한다. 이런 류사성으로 하여 서로 다른 파일들을 어떻게 찾는가 하는 문제가 제기된다. 둘째로 행동파일은 .dt 로 끝나고 서술파일은 수행될수 있는것으로 되여야 한다. 만일 어느 하나가 틀리면 행동은 작용하지 않거나 나타나지 않는다.

많은 입력를 피하기 위한 쉬운 방법은 이미 존재하는 정의된 파일을 복사하고 새 조종으로 필요한것을 삽입한다음 변경시키는것이다. 조종규정목록을 아래로 이동시키는것은 정면판이 표시될 때 왼쪽에서 오른쪽으로 이동시키는것과 같다. (POSITION_HINTS 의 값은 매 정의마다 증가된다.)만일 정면판에서 날자의 오른쪽에 조종을 기입하려고 한다면 date 아래의 행에 조종을 삽입하고 POSITION_HINTS 3 행을 첨가한다. 만일 날자의

왼쪽에 조종을 기입하려고 한다면 date 우의 행에 조종을 삽입하고 POSITION_HINTS 1 을 첨가한다.

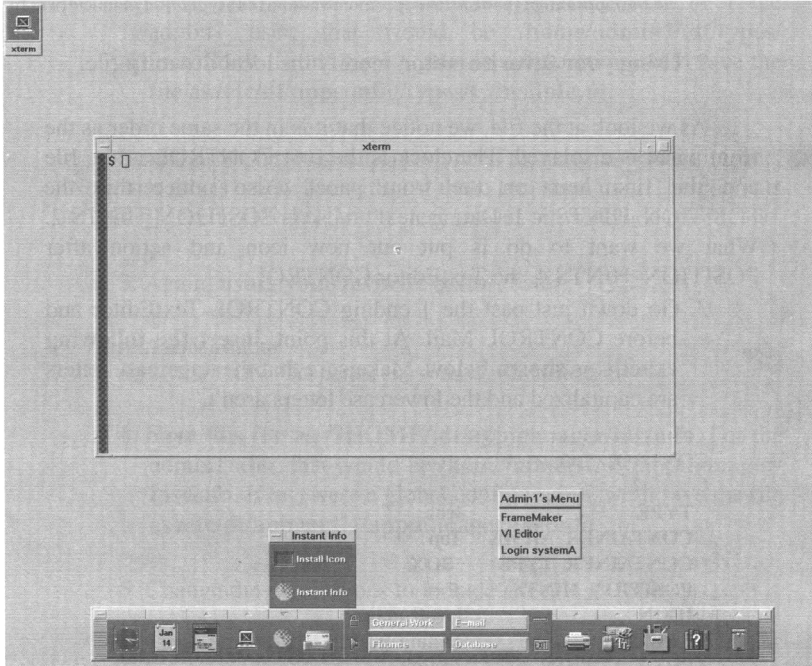



그림 15-6. 작업공간안내와 새 부분판 및 행동

새 조종에 대한 정의는 조종정의목록에서 어디에나 배치될수 있다. POSITION_HINTS 행은 새 위치에서 우연적인 충돌을 피하게 한다. 이것은 이미 존재하는 규정된 복사와 필요 없는 입력을 피하게 한다. 따라서 이것은 입력오유를 줄일수 있게 한다. 여기서 대괄호 "}"를 포함시키는것을 잊지 말아야 한다.

다른 서체들의 리용

이미 선택된 CDE 의 서체들외에 다른 서체를 리용할것을 요구할수 있다. CDE 환경을 통하여 체계범위에서 리용될수 있는 서체들을 취하려면 Style Manager 의 서체대화칸에 나타나도록 그 서체들을 /etc/dt/app_defaults/dtstyle 에 넣어야 한다. 특이한 X 의뢰기응용프로그람에서만 리용할 서체를 선택하려면 응용프로그람이 파일 app_defults 에서 서체를 규정해 주어야 한다. 이것은 Style Manager 에서 서체들을 무시하게 한다. 서체대화칸에는 최대로 7 개 서체들이 있을수 있다. /etc/dt/app_defaults/Dtstyle 에서 Dtstyle*NumFonts 의 값을 재설정하여 이 개수를 더 작게 조절할수 있다. 그러나 7 개보다 더 큰 개수로 증가시킬수는 없다. Dtstyle 구성파일에서 Font Dialog 부분에는 7 개 체계서체자원들과 7 개 사용자서체자원들이 있다. 여기서도 7 개 체계서체와 7 개 사용자서체보다는 작게 가질수 있으나 더 많이 가질수는 없다. 특수한 응용프로그람에서 서체를 규정하려면 응용

프로그람이 파일 app_defaults 에서 *FontList 자원을 리용하여야 한다. 개별적인 사용자들의 요구에 맞게 서체자원을 변경시키려면 이미 앞에서 서술된 정면판의 변경부분에서와는 달리 EditResources 행동을 리용하여야 한다.

가입통보의 변경

CDE 가 가지고 있는 우점의 하나는 많은 부분들을 변경시킬수 있다는것이다. 즉 개별적인 가입계산번호 혹은 체계를 완전히 개별화할수 있다는것이다. 가입화면을 개별화하면 체계이름, 기관이름, 시작통보를 볼수 있다. 이 변경은 파일 Xresources 에서 진행한다.

1) 파일 dtwm.fp와 dtwmrc에서 이미 한것처럼 체계파일을 /usr/dt/config/C로부터 /etc/dt/config/C에 복사하여야 한다. 즉

   cp/usr/dt/config/C/Xresourxes /etc/dt/config/C/Xresources

2) 편리한 편집기에 가서 편집할수 있도록 파일 Xresources 을 연다.
3) GREETING 령역에로 간다. 여기서 다음과 같은 행들을 볼수 있다.

```
!!Dtlogin*greeting.labelString:           Welcome to %LocalHost%
!!Dtlogin*greetingpersLabelString:        Welcom %s
```

첫번째 행은 시동가입화면에서의 통보문을 보여 준다. 기관에서 환영한다는 통보를 ABC, Inc.로 다음과 같이 변경시켜 보자.

1) 설명표시를 제거한다. 대부분의 리용되는 쉘스크립트와는 달리 파일 Xresources 은 설명을 지적하는데 두개의 기호 "!!"를 제거한다.
2) 다음에 Welcome to % LocalHost%를 변경시킨다. 변수 %LocalHost%는 가입화면에서 체계이름으로 교체된다. 이 행은 다음과 같다.

```
Dtlogin*greeting.labelString:        ABC, Inc, Welcom You to %LocalHost%
```

3) 다음으로 이 체계를 리용하는 기관을 포함시키기 위하여 두번째 행을 finance 로 변경시킨다. 이 두번째 행은 암호가 예고될 때 표시될 정보를 보여 준다. 변수 %s 는 사용자이름이다. 이 행은 다음과 같다.

```
Dtlogin*greetingpersLabelString:        The Finance Department Welcomes %s
```

4) 파일을 보관한다.
5) 이제 가입을 취소하고 다시 가입을 할수 있다. 이때 가입화면에서 변경된것들을 볼수 있을것이다. 가입취소의 행동과 거꾸로 그 행동을 하기때문에 그 파일을 재적재

할 필요는 없다.

가입그림의 변경

만일 가입화면에 새 그림을 추가하는 어느 한 방법을 알고 있다면 쉬울것이다. 그 파일은 bitmap(.bm) 혹은 pixmap(.pm)파일로 되여야 한다. 비트매프파일은 흑백색이고 pixmap 파일은 천연색을 가진다. 여기서 그림의 다른 종류들(.gif 와 .jpg 형식)을 리용하지만 이 책에서는 표시되지 않는다. 좋은 새 방법은 다른 체계 혹은 인터네트로부터 리용할 파일을 받아 들일수 있는것이다. 단순히 관찰하기 위하여 체계에 이미 준비된 파일을 리용할수 있다. 실례로 생일과자를 보여 주는 비트매프파일은 HP-UX 의 /usr/lib/X11/bitmaps 에서 찾을수 있다. 또한 꽃그림 Grand Canyon 을 끌어다 놓고 인터네트로부터 노래를 끌어다 체계가입화면에 놓는다.

1) 한번 더 적당한 편집기를 열고 /etc/dt/config/C/Xresources 를 변경시킨다.
2) MISC 령역으로 간다. 여기에서 다음과 같은 행을 찾을수 있다.

```
!!Dtlogin*logo*bitmapFile:          <bitmap or pixmap file>
```

3) 앞에 있는 설명문기호 "!!"를 제거한다.
4) bitmap or pixmap file 을 완전경로위치를 리용하여 비트매프파일의 이름으로 교체한다. 이 행은 다음과 같다.

```
Dtlogin*logo*bitmapFile:          /usr/lib/X11/bitmap/cake.bm
```

5) 파일을 보관한다.
6) 가입을 취소하고 다시 가입한다. 그러면 가입화면에서 생일과자를 볼수 있다. 가입취소행동과 거꾸로 이 과제를 수행하여 가입하였으므로 파일을 재적재할 필요는 없다. 지금까지 CDE 에서 단순한 개별화의 방법을 보았다. 이러한 지식을 가지면 실지로 능률적이면서 흔히 작업하는데 필요한 자기의 CDE 환경을 만들수 있다.

CDE-개선된 항목

X, Motif, CDE 사이의 관계

X, OSF/Motif 와 CDE 는 가능한 **프레임**(Frame)작업기술이다. 함께 취급할 X, Motif, CDE 는 조작체계와 하드웨어환경이 우에 있는 세개의 도형층을 조성한다. 조작체계층이 사용자방향인 기호기초대면부로부터 말단사용자를 보충하는 지속적인 렬로서 GUI 층은

여러가지 기능을 쉽게 리용할수 있게 하여 준다.

X Window 체계

X Window 체계는 다음과 같은것으로 구성되여 있다.

- Xlib – 프로그람작성창문조작을 위한 낮은 수준의 서고, 선그리기와 본문배치와 같은 그라프능력, 화면표시조종, 마우스, 건반입구조종과 응용프로그람에 대한 망의 일관성.
- Xt Intrinsics – 프로그람작성 widget 와 gadget 들의 높은 수준의 서고(안내, 흐름띠, 누르기단추와 같은 도형조종성분들).
- Display servers – 도형입력과 출력을 관리하는 화면장치당 한개의 장치규정프로그람들.
- Interclient communication conventions(ICCC) – X 의뢰자프로그람이 어떻게 다른 매개의 의뢰자프로그람과 통신할것인가에 대한 표준을 규정하는 편람.
- Configuration files - 시동할 때 기정가입기간(sys.x11start)을 규정하고 X 환경의 조성에 리용될 자원들의 값들을 규정할 구성파일 (sys.Xdefaults).

이와 같이 X 는 망지향의 도형부분, 의뢰기/봉사기, 분산된 계산모형에 기초한 표준화를 제공한다. Xlib 와 Xt Intrinsics 에 대한 지식은 X 와 Motif 수준에서의 프로그람작성에 중요하게 리용된다. 그러나 체계관리자에 대한 현시봉사기작업과 X 의뢰기응용프로그람들은 ICCC 가 친절하기때문에 X 층에 대하여 깊이 연구할 필요는 없다. CDE 는 X 에 대한 기술적하부구조의 세부를 필요한만큼 볼수 있게 하며 따라서 사용자는 CDE 의 적당한 모형개발에만 집중할수 있게 한다.

Motif

Motif 는 다음과 같은것으로 구성되여 있다.

- mwm window manager – Motif 에 기초한 창문프레임들, 창문관리, X 환경에서 작업공간안내를 제공하는 수행할수 있는 프로그람
- Motif widget toolkit – widgets 와 gadgets 의 높은 수준의 서고, 사용자환경조종에 리용될 도형성분들
- Motif style guide – Motif 출현과 프로그람작성자의 행동을 정의하는 편람
- Configuration files – 작업공간안내와 건반과 단추의 묶기를 위한 모형정보를 포함하는 system.mwmrc 파일. 창문관리자의 자원들은 /usr/lib/X11/app-defaults 등록부에서 mwm 으로 있다.

Motif 는 말단사용자를 위한 창문관리자, 응용프로그람개발자들을 위한 widget 도구,

개발자의 설계와 Motif 에 적응된 응용프로그람구축을 돕기 위한 양식편람을 제공한다. X 와 같은 체계조직조직자는 Motif 에 대한 하부구조를 필요한만큼 볼수 있게 하며 따라서 CDE 환경파일들에 대한 개발에 집중할수 있게 한다.

CDE

CDE 는 다음과 같은것들로 구성되여 있다.

- Workspace Manager – Motif 에 기초한 창문프레임, 창문관리, 작업공간안내, 정면판을 제공하는 수행될수 있는 프로그람
- File Manager – 직접 조작으로 파일과 등록부의 그림기호를 관리하는 프로그람
- Style Manager – 작업공간의 색파 서체들과 같은 CDE 환경의 요소들을 조종하는 대화칸의 포함자
- Help Manager – 이 프로그람은 CDE 성분들에서 문맥에 따르는 도움말본문을 제공한다.
- Login Manager – 가입과 암호확증을 조종하는 전문적인 프로그람
- Session Manager – 사용자가입기간에 대한 통보의 보관과 복구를 조종하는 관리자
- Application Manager – CDE 환경에서 응용프로그람들의 추적을 유지하고 등록하는 관리자
- Configuration Files – 대부분의 많은 처리를 피하게 할수 있는 큰 묶음파일들

또한 CDE 는 많은 말단사용자들에게 필요한 기본적인 응용프로그람들을 제공한다. 일반적으로 CDE 는 사용자들 혹은 체계관리자가 작업하는데 필요한 쏘프트웨어도구들을 통합할수 있는 도형환경을 제공한다.

X, Motif, CDE 의 구성파일

X, Motif, CDE 는 모두 그 형태와 행동의 형식을 규정하는 구성파일들을 리용한다. 배경색, 건반초점방법, 의뢰자장식과 같은 형태와 행동의 요소들은 구성파일에서 값으로 조종될수 있는 자원들이다. X, Motif, CDE 에서 자원이라는 단어는 특수한 의미를 가진다. 그것은 자연부원의 자원이나 일반적인 체계자원들을 말하는것이 아니라 형태와 행동에 대한 특정의 요소들을 의미한다. 실례로 전경자원, 건반초점방법의 자원, 의뢰자의 장식자원을 들수 있다. 여기서 전경색은 흑색이고 건반초점방법은 명백하며 의뢰자장식은 제목띠에서 색의 흐름으로 될수 있다. 이것들은 다음과 같은 적당한 구성파일에서 나타난다.

```
*foreground:              black
*keyboardFocusPolicy:     explicit
*clientDecoration:        +title
```

구성파일의 이 자원들은 요구되는 작용범위(체계범위 혹은 개별적사용자)와 리용되는 도형대면부의 수준(X, Motif, CDE)에 따라 다르게 나타난다.

X구성파일

X 창문체계는 다음과 같은 구성파일들을 가진다.

sys.x11start
sys.Xdefaults
system.mwmrc
X*screens
X*devices
X*pointerkey

이미 언급한것처럼 이 파일들은 등록부 /usr/lib/X11 에 있다. 그러나 많은 체계들은 이 등록부를 제거하고 체계의 어디에나 많은 X 관련파일들을 이동시킬수 있다. 추가적으로 매 X 의뢰기응용프로그람은 이미 언급한것처럼 등록부 /usr/lib/X11/app-defaults 에 있는 고유한 app-defaults 구성파일을 가지고 있다. 비록 6 개 파일들은 우에서 모형화되였지만 다중표시화면들(X*Screens), 다중입력장치들(X*devices), 건반지시자표시(X*pointerkey)를 위한 작업장소를 모형화하지 않는한 보통 sys.X11start, sys.Xdefaults, system.mwmrc 만으로 작업하는것이 필요하다.

파일 sys.X11start 은 CDE 가 출현하기전에 X 와 X 의뢰기의 시동에 리용될 스크립트이다. 체계관리자 혹은 지식이 있는 사용자들은 X 의뢰기들이 자동적으로 결합되도록 sys.X11start 를 변경시킨다. 파일 sys.Xdefaults 은 시동된 X 를 리용하여 여러가지 형태와 행동자원들에 대한 값들을 읽는다. X 환경과 의뢰자들의 요구에 맞는 sys.Xdefaults 의 변경은 고유한 모양과 행동을 가지게 한다. system.mwmrc 는 작업공간안내와 단추, 건반묶기모형을 포함한다. system.mwmrc 도 Motif 방안에 따라 달라 진다.

Motif구성파일

Motif 는 X 목록에 한개의 새 구성파일인 tem.mwmrc 만을 추가하였다. 이미 언급한것처럼 이 파일은 /urs/lib/X11 에 X 구성파일과 함께 있다. 실제로 이 파일은 새것이 아니고 Motif 환경에서 system.mwmrc 로 교체된 system.mwmrc 의 Motif 방안의 파일이다. 망과 의뢰기사이의 통신은 도형사용자대면부로 표준화되였기때문에 Motif 는 대부분 PC 에 기초한 GUI 의 기초를 형성하였으며 X 는 IBM 의 체계응용프로그람방식공동사용자입장(SAACUA)에서 원래 정의된 표준적인 형태와 행동을 표준으로 취하였다. 그리하여 누르기단추와 흐름띠는 이미 정의된 보기와 행동을 진행하며 이것들을 두번 선택하면 항상 기정행동을 진행하게 된다. 프로그람작성자의 관점으로부터 Motif Widget Toolkit 는 X 에서 좀 미약하던 프로그람작성에 많은 수단들을 제공해 준다. 사용자나 체계조직관리자의

관점으로부터 Motif의 사용자환경은 mwm Window Manager가 motif 창문관리자로 교체된 것을 제외하고는 X환경과 같다.

CDE구성파일

CDE의 모형화를 위해서는 80개이상의 파일들을 리용할수 있다. 이미 언급한것처럼 이 파일들은 /usr/dt 등록부에 있다. 다른 파일들과 같이 이 목록으로부터 이동시키려면 다음과 같이 할수 있다.

- 환경자체가 대립되는 CDE 응용프로그람들을 모형화한다.
- 개별적파일들은 고유한 정의를 조성하지만 결코 변경되지 않는 기정행동과 자료형을 규정한다.
- CDE의 작업파일들은 있으며 개별화되지 않는다.
- 여러가지 쉘환경을 포함하여 CDE에 있는 UNIX, X, Motif 환경의 모형화와 적당히 관련되여 있다.

그러면 CDE는 표 15-1에서 보여 주는것처럼 대체로 19개의 구성파일들을 가진다.

표 15-1 CDE 구성파일들

*.Xauthority	* sys.font	* Xresources
*.Xdefaults	* sys.resources	* Xservers
*.dtprofile	* sys.sessions	* Xsession
*dtwm.fp	* Xaccess	* Xsetup
*dt.wmrc	* Xconfig	* Xstartup
*sys.dtproftle	* Xfailsafe	
*sys.dtwmrc	* Xreset	

여기서 19개 구성파일들이 지적되였으나 그 외에도 더 많은 파일들이 있다. 많은 파일을 변경시킬 필요는 없고 어느 한 파일을 변경시키면 다른 파일에서의 값을 무시할수도 있다. 만일 정면판을 변경시켜야 할 필요가 제기되고 개별행동과 자료형에 대한 체계범위파일 *.dt 혹은 dtwm.fp를 주기적으로 변경시키려면 이 파일에 대한 깊은 리해를 가지는것이 필요하다. CDE 모형화는 일정한 준비와 파악이 없이 변경을 시작하지 말아야 한다. 모든 CDE 환경파일들을 변경시킬 때에는 먼저 그것을 인쇄하는것이 좋다. 표 15-2에서는 CDE 구성파일의 내용과 그 영향의 작용범위에 따라 CDE구성파일들을 분류해 주고 있다.

표 15-2 CDE 구성파일의 영향

구성파일의 본질	체계범위영향	사용자개별영향
환경변수들	sys.dtprofile Xconfig Xsession	.dtprofile
외모와 행동자원들	sys.resources Xconfig Xresources sys.fonts	.Xdefaults
파일형 및 행동정의	misc *.dt files	user-prefs.dt
가입할 때 의뢰기시작	sys.sessions Xstartup Xsession Xreset Xfailsafe	.xsession sessionetc
작업공간관리자 및 정면판	sys.dtwmrc dtwm.fp	dtwmrc user-prefs.fp
의뢰기 봉사기 및 입장	Xaccess Xservers	.Xauthority

파일 sys.dtwmrc 은 체계수준에서 작업공간관리자의 모형을 조종한다. 이것은 다음과 같은것들을 포함한다.

Workspace Menu　마우스지시자가 작업공간의 배경막우에 있을동안 마우스의 셋째 단추를 누를 때 표시되는 안내이다.

Button Bindings　마우스지시자가 특이한 령역(프레임, 그림기호, 창문, 뿌리) 에 있을동안 특정한 마우스단추를 누르거나 마우스지시자가 해방될 때 나타날 행동을 정의한다.

Key Bindings　마우스지시자가 특이한 령역(프레임, 그림기호, 창문, 뿌리)에 있을동안 특수건 혹은 건반렬을 누를 때 나타날 행동을 정의한다.

X 혹은 Motif 의 구성파일과는 달리 sys.dtwmrc 는 다음과 같은 모형요소들을 조종할수 없다.

Front Panel　6 개 단추인 작업공간절환을 포함하여 흔히 리용되는 도형조종들과 공통적으로 창조되는 지시기를 포함한 작업공간의 아래에 보통 나타나는 창문이다.

Slideup Subpanels　많은 화면공간을 리용하지 않고 더 많은 기능을 제공하기 위하여 정면판의 여러 위치에 표시될수 있는 내리펼침차림

표들이다.

지나치게 크고 복잡한 구성파일들을 피하기 위하여 이 요소들은 CDE 에서의 고유한 구성파일인 dtwm.fp 에서 분할되여 있다. 작업공간의 개수와 작업공간절환들의 정돈과 같은 정면판의 일부 모형요소들은 sys.resources, dt.resources, .Xdefaults 파일들의 자원들을 통하여 조종된다. 다른 작업공간관리자의 구성파일들과 같이 sys.dtwmrc 는 사용자의 홈등록부에 복사될수 있으며 실지 $HOME/.dt/에 dtwmrc 로 복사되고 sys.dtwmrc 의 체계범위모형을 벗어 나 사용자환경을 개별화하기 위하여 변경된다.

파일 sys.resources 은 한번 변경될수 있는 파일들중의 하나이고 다시 변경될수 없다. 파일 dt.resources 은 변경되고 무시될수 없는 파일들중의 하나이다. 파일 .Xdefaults 은 사용자가 변경시킬수 있는 파일이다. 파일 sys.resouces 은 새로운 사용자가 CDE 에 제일 처음으로 가입할 때 작용할 임의의 기정이 아닌 자원들을 포함한다. 실례로 체계관리자는 사용자들에게 이미 이름 지어 진 작업공간, 규정된 색들, 특이한 서체들, 일정한 위치의 응용프로그람창문을 가진 CDE 정면판을 가질것을 요구할수 있다. 첫번째로 가입한 후에 sys.resources 는 dt.resources 의 자원에 의하여 무시된다. 파일 dt.resources 은 $HOME/.dt /sessions/current 에 있고(혹은 홈가입기간이 복구될 때에는 $HOME/.dt/sessions/home 에 있다.) CDE 에 의하여 자동적으로 조성된다. 그것을 CDE 의 작업파일로 고려하고 잊어 버릴수 있다. 파일 .Xdefaults 은 말단사용자가 개별 CDE 환경에서 규정한 X 자원들의 목록화를 포함한다.

sys.resources, dt.resources, .Xdefaults 는 자원의 목록과 그 값들을 포함한다. 파일 sys.sessions 은 CDE 에 첫번째로 새로운 사용자가 가입할 때 시동된 의뢰기들을 조종한다. dt.resources 가 sys.resources 에 있는것처럼 파일 dt.session 은 sys.sessions 에 있다. 이 파일은 특이한 응용프로그람시동을 위한 CDE 모형화에 리용된다. sys.sessions 에서 이 응용프로그람들을 규정할수 있다. 첫번째로 새로운 사용자가 가입할 때 CDE 환경은 규정된 의뢰기들을 포함한다. 가입을 취소하여 첫 가입기간이 끝날 때 남아 있는 의뢰기들은 CDE 의 $HOME/ .dt/sessions/current 에 기억된다. (홈가입기간이 복구될 때 남아 있는 의뢰기들은 $HOME/.dt/ sessi-ons/home 에 기억된다.)

sys.dtprofile 은 새로운 사용자가 홈등록부에 처음으로 가입할 때 자동적으로 .dtprofile 로 복사되는 **본보기파일**(Template File)이다. sys.dtprofile 은 CDE 환경에서 .profile 혹은 .login 으로 교체된다. (.profile 혹은 .login 은 DTSOURCEPROFILE = true 의 앞에 있는 설명기호 "#"를 제거함으로써 .dtprofile 의 원천파일로 될수 있다.)

파일 .dtprofile 은 기호기초환경의 .profile 혹은 .login 에 있는 개별적환경변수들을 포함한다. CDE 의 그라프환경으로부터의 영향을 피하기 위하여 말단 I/O 지령들은 .dtprofile 을 리용한다. CDE 가입관리자인 dtlogin 은 다음 환경변수들을 기정값들로 미리 설정한다.

DISPLAY	국부적표시장치이름
EDITOR	기정본문편집기
HOME	/etc/passwd 에서 규정된것과 같은 사용자의 홈등록부
KBD_LANG	건반에 대응되여 있는 현재언어
LANG	현재 NLS 언어

LC_ALL	LANG 의 값
LC_MESSAGES	LANG 의 값
LOGNAME	/etc/paawd 에서 규정된것과 같은 사용자의 가입이름
MAIL	전자우편의 기정파일(/var/mail/$USER)
PATH	파일들과 응용프로그람들을 찾을 기정등록부들
USER	사용자이름
SHELL	/etc/passwd 에서 규정된것과 같은 기정쉘
TERM	기정말단모방
TZ	작용하고 있는 시간지역

이 기정값들을 가지는 변수들은 매 사용자의 .dtprofile 에 포함되여 있다. 추가적환경변수들은 작업문맥에 따라 사용자환경의 모양에 필요한만큼 추가될수 있다. 임의의 말단 I/O 를 동반하는 지령의 리용에 주의해야 한다. .dtprofile 과 같이 Xsession 는 사용자환경변수를 설정할 쉘스크립트이다. Xsession 에서 환경변수들은 체계범위에서 리용된다. .dtprofile 에서 환경변수들을 사용자의 개별환경에서만 리용된다. 가입관리자는 X 봉사기가 시동된 후에 Xsession 를 수행하기때문에 Xsession 에서의 변수들은 X 봉사기에서 리용될수 없다. Xsession 에서 변수들은 보통 다음과 같이 설정된다.

EDITOR	기정본문편집기
KBD_LANG	건반에 대응되여 있는 현재언어(보통 $LANG 의 값으로 설정한다.)
TERM	기정말단모형
MAIL	/var/mail/$USER 인 전자우편의 기정파일
DTHELPSEARCHPATH	CDE 도움말파일을 찾는 위치들
DTAPPSEARCHPATH	CDE 응용프로그람관리자로 등록될 응용프로그람관리자를 찾는 위치들
DTDATABASESEARCHPATH	추가적행동과 자료형정의를 찾을 위치들
XMICONSEARCHPATH	추가적그림기호를 찾을 위치들
XMICONBMSEARCHPATH	우에서와 같다.

실례로 여러가지가 혼합된 작업장소와 어느 한 지점에 있는 X 말단묶음에 대한 어떤 체계관리자가 있다고 하자. 대부분 사용자들은 어떤 본문편집기에 습관되여 있다. 어떤 사용자들은 vi, 다른 사용자들은 emacs 를 좋아하고 이 사용자들은 dmx 가 없이도 불편을 느끼지 않는다.

사용자들이 좋아하는 본문편집기를 매 사용자에게 보장하기 위한 쉬운 방법은 개별적인 .dtprofile 에서 적당한 값으로 EDITOR 변수를 재설정하는것이다.

Xconfig 는 dtlogin 의 행동을 조종할 자원들을 포함하며 또한 임의의 다른 dtlogin 구성파일이 조성될 위치도 규정해 준다. 파일 Xconfig 는 체계범위에 기초하여 작업하며 이 파일들중 하나를 변경한다음에 잊어 버릴수도 있다. 체계에 가입할동안 Xconfig 를 수행

할 때 여러개의 CDE 구성파일들은 Xaccess, Xservers, Xresources, Xstartup, Xsession, Xreset, Xfilesafe 를 참조한다. Xconfig 와 같이 이 파일들이 CDE 를 설정될 때 일단 변경된다음 망위상이 변경되지 않으면 그것을 다시 변경할 필요는 없다.

이름을 보고 알수 있는것처럼 Xaccess 는 원격적표시접근조종파일이다. Xaccess 는 국부적콤퓨터에 XDMCP 접속접근을 허용하거나 완성되지 않은 주콤퓨터이름의 목록을 포함한다. 실례로 X 말단의 가입봉사를 요구할 때 dtlogin 은 매번 봉사가 담보되는가를 결정하기 위하여 파일 Xaccess 를 리용한다. 파일 Xservers 의 1 차적리용은 dtlogin 의 관리에 반응할수 있는 국부적체계에서의 표시화면을 목록화하는것이다. dtlogin 은 파일 Xservers 을 읽고 여기서 목록화된 매 표시를 위한 X 봉사기를 시동시킨다. 그다음 봉사기를 관리하고 가입화면을 표시하기 위하여 후손프로쎄스 dtlogin 을 시동시킨다. dtlogin 은 국부적으로만 작업한다는것을 지적해 둔다. dtlogin 은 원격체계 혹은 X 말단에서 X 봉사기를 시동시킬수 없다. 원격표시봉사기들에 대하여 일부 다른 기구는 봉사기시동에 리용되며 다음에 dtlogin 으로부터 가입화면을 요청하기 위하여 X 표시관리조종규약(XDMCP)을 리용한다.

파일 Xservers 는 시동시키는데 일정한 시간을 보내는 파일들중의 어느 한 파일이고 망위상이 변경되지 않으면 다시 변경될수 없다. 언제 Xservers 를 리용하는가? Xservers 는 표시가 기정모형과 일치되지 않을 때 리용된다. 기정모형은 매 체계가 단일한 비트매프표시를 가지는 체계조작대라고 생각할수 있다. X 말단들, 다중표시장치들, 다중화면들, Starbase 응용프로그람들은 모두 파일 Xservers 에서 모형행들을 요구한다. 파일 Xservers 는 가입화면의 형태와 행동을 조종하는 자원들의 목록을 포함한다. CDE 가입에 기관의 가입을 대입하고 서체와 색을 변경한 후에 Xresource 를 다시 변경할수 없다. (상사가 가입을 변경시키지 않는다면)

Xstartup 은 여러개의 환경변수들을 리용하는 가입관리자에 의하여 수행되는 체계범위의 구성파일이다.

DISPLAY	국부적표시이름
USER	사용자의 가입이름
HOME	사용자의 홈등록부
PATH	Xconfig 에서 systemPath 자원의 값
SHELL	Xconfig 에서 systemShell 자원의 값
XAUTHORITY	관권허락의 접근을 위한 파일
TZ	국부적시간대

Xstartup 는 스크립트를 수행시킬수 있고 체계범위에 기초한 의뢰기들을 시동시킬수 있기때문에 sys.sessions 와 류사하다. 차이점은 Xstartup 이 뿌리로서 수행된다는것이다. 그리하여 Xstartup 에서 변경은 설치된 파일체계와 같은 행동에 작용한다. Xreset 는

Xstartup 의 체계범위에서 리용되는 스크립트이다. 그것은 뿌리로서 수행되며 Xstartup 을 조종하지 못하게 한다. Xfailsafe 는 표준 failsafe 가입기간의 개별화에 대한 통보를 포함한다. failsafe 가입기간은 가입과 가입기간구성파일에서의 오유로 나타난 CDE 가입기간을 수정할 방법을 준다.

Xfailsafe 는 사용자가 리용할수 없는 일부 경우들도 있으나 좋은 개별화를 리용하여 쉽게 할수도 있다. sessionetc 는 사용자의 등록부.dt/sessions 에 있고 사용자의 CDE 가입기간을 개별화한다.

sessionetc 는 sys.sessions 와 같은 추가적 X 의뢰기의 시동을 조종하거나 체계범위와 달리 사용자범위로 리용된다. dt.session 은 사용자마다 의뢰기들을 시동시키며 그 의뢰기들은 기정 혹은 표준가입기간의 의뢰기로 된다. dt.session 은 .dt/session 에 있는 .dt/session/current.sessionetc 에 있고 자동적으로 복구되지 않는 의뢰기들만 포함하여야 한다.

대표적으로 WM_COMMAND 를 설정하지 않는 의뢰기들이 있는데 가입기간관리자는 그것을 보관하거나 복구할수 없다. 그것들은 sessionetc 에서 재시동하는것이 필요하다. 파일 sys.font 는 체계범위에서 리용되며 기정가입기간의 서체모형을 포함한다. 이 기정서체들은 잘 만들어 져 있으므로 파일 sys.font 는 변경시킬수 없다. 그러나 체계범위를 리용하는 다른 서체들과 혼합하여 사용할 필요가 있는 경우에는 이것을 변경시킬수 있다. sys.font 에서 언급된 서체자원들과 값들은 파일 /usr/dt/app_defaults/C/dtstyle 에서 규정된 기정서체자원과 정확히 일치되여야 한다. CDE 는 CDE 행동과 자료형정의를 규정하는 파일들의 묶음이다. 이 모든 파일들은 *.dt 와 같이 확장자 dt 로 되여 있다. *.dt(탁상화면에서는 dt)는 자료형과 행동정의를 둘 다 포함한다. 기정파일 *.dt 들은 /usr/dt/appconfig /types/C 에 있고 체계범위로 리용된다. 류사하게 user_prefs.dt 를 /usr/dt/appconfig/types/C 에서 복사하여 이것을 개인사용자수준에서 리용한다.

권한부여수법(Authorization Mechanism)을 리용하는 의뢰기를 봉사기에 접속하기 위하여 권한부여통보가 필요되는데 파일 .Xauthority 은 이 통보를 포함한다. 이것은 사용자규정의 구성파일이다.

CDE구성파일의 위치

CDE 가 어떤 구성파일을 어디에서 찾아 보아야 하는가는 원리적으로 파일이 무엇을 모형화하였으며 파일의 작용범위가 어느정도인가 하는 구성파일의 본질에 의존한다.

표 15-3 은 파일내용의 본질에 기초하여 체계와 사용자구성파일의 위치를 보여 준다. 매개 기정체계범위의 파일위치들은 표 15-3 에서 목록화되였으며 대응되는 위치는 개별적체계범위의 구성파일에 있다. 이 개별적파일들은 등록부 /etc/dt 의 적당한 부분등록부에 들어 있다. 기본절차는 /usr/dt/something 으로부터 /etc/dt/something 에 개별화에 필요한 파일들을 복사하고 그다음 복사된 파일을 변경하는것이다. 실례로 Xresources 에서 기정가입을 변경시키기 위하여 /etc/dt/config/C/Xresources 에 /usr/dt/config/C/Xresources 를 복사하고 /etc/dt/config/C/Xresources 을 연다음 변경시킨다.

표 15-3 CDE 체계와 사용자구성파일들

구성파일의 본질	체계범위 영향	사용자개별 영향
환경변수들	/usr/dt/config/	$HOME/
외모와 행동자원들	/usr/dt/config/C /usr/dt/app-defaults/C	$HOME/.dt/ $HOME/.dt/sessions/current/ $HOME/.dt/sessions/home/
파일형 및 행동정의	/usr/dt/appconfig/ types/C	$HOME/.dt/types
가입할 때 의뢰기시작	/usr/dt/config/ /usr/dt/config/C	$HOME/.dt/session/ $HOME/.dt/session/current/ $HOME/.dt/session/home/
작업공간관리자	/usr/dt/config	$HOME/.dt/

이것은 중요한 문제이다. /usr/dt 에 배치된 파일들은 CDE 체계파일들로 되며 갱신할때 덧씌여 질수 있다. 여기서 임의의 개별화는 이미 있던것을 지워 버린다. /etc/dt 와 그 부분등록부에서 체계범위의 구성파일에 대한 모든 변경을 진행하여야 한다.

구성파일의 수행결과가 어떻게 나타나는가

지금까지 발표된 문헌으로부터 CDE 구성파일들을 면밀한 계획을 가지고 변경시켜야 한다는것을 알수 있다. 모형화할 요소와 요구되는 작용범위는 어떤 구성파일을 변경시킬것인가를 결정할수 있게 한다. 실례로 만일 어떤 환경변수를 설정하려고 한다면 4 개 구성파일들 sys.dtprofile, Xconfig, Xsession, .dtprofile 중에서 하나를 선택해야 한다. 그러나 만일 특정한 사용자에만 적용되는 환경변수들을 설정하려고 한다면 파일 .dtprofile 에서 직접 필요한 선택을 할수 있다.

이제 남아 있는 조작은 CDE 가 구성파일들을 읽는 순서를 리해하는것이다. 모형의 요소(환경변수, 자원, 행동, 자료형)가 두번 규정되지만 다른 값을 가진다면 리용될 정확한 값은 명백히 요구되여야 하며 정확하지 않은 값은 무시되여야 한다. 이때 다음과 같은 규칙이 적용된다.

- 환경변수에 대하여서는 마지막에 규정된 값이 리용된다.
- 자원에 대하여서는 마지막에 규정된 값이 리용된다. 그러나 이것은 일련의 특성으로 반영된다. 그리하여 emacs*foreground 는 자원들이 맞다들린 순서에 무관계하게 emacs 의뢰기의 *foreground 에 우선권을 준다.
- 행동에 대하여서는 첫번째로 규정된 값을 리용한다.
- 자료형에 대하여서는 첫번째로 규정된 값을 리용한다.

표 15-4 는 CDE 가 그 구성파일들에서 모형요소들이 여러번 규정되였을 때 리용될 규정에 대한 실례를 보여 준다.

표 15-4 CDE 는 모형에서 무엇을 리용하는가

모형요소	리용된 요소
자원	마지막규정 혹은 가장 많은 규정
환경	마지막으로 맞다들린 환경
행동	첫번째로 맞다들린 환경
파일형	첫번째로 맞다들린 파일

작업범위에 따라 사용자구성파일은 체계범위구성파일을 무시한다. 체계범위구성파일들의 우선권순서를 보면 /etc/dt 에서 파일들은 /usr/dt 에서의 파일보다 더 높은 우선권을 가지며 따라서 **전역적전용구성**(Global Custom Configurations)은 CDE 의 기정모형보다 우선권이 높다. $HOME/.dt 파일들은 /etc/dt 에서의 파일보다 우선권이 높다. 자원에 대하여 GUI 의 형태와 행동을 규정하는데 리용되는 요소들은 다음과 같은 우선권에 따라 값을 설정한다.

1) Command line - 지령행으로부터 의뢰기가 시동될 때 지령행에서 렬거된 선택항목들은 제일 높은 우선권을 가진다.
2) xresource, .Xdefaults, dt.resource, sys.resource - CDE 가 시동될 때 그 가입기간에서 리용될 X 자원들의 값을 결정하기 위하여 그 자원구성파일들을 읽는다.
3) RESOURCE MANAGER - RESOURCE MANAGER 속성에 이미 있던 자원들은 시동되는 응용프로그람에서 리용된다.
4) app_defaults - 표준자원값과 다른 기정자원값을 규정한다.
5) built_in defaults -고정적으로 코드화된 기정자원들은 가장 낮은 우선권을 가진다.

특정의 자원규정은 일반자원규정보다 우선권이 높다. 실례로 본문입구령역에 일정한 서체를 요구한다고 가정하자. 개별적 .Xdefaults 파일에서 *fontreset 자원을 정확히 규정하여야 하며 app_defaults 파일에서만 XmTextFontList 를 무시하도록 한다. 비록 app_defaults 는 .Xdefaults 보다 더 낮은 우선권이지만 자원규정은 우선권을 가지도록 규정된다. 환경변수에 대하여 CDE 는 다음과 같은 우선권에 따라 값을 설정한다.

1) $HOME/.dtprofile - 사용자규정변수는 제일 높은 우선권을 가진다.
2) /etc/dt/config/C/Xsession- 개별체계범위의 변수들은 X 봉사기로는 읽지 못한다.
3) /etc/dt/config/C/Xconfig - 개별체계범위의 변수들은 X 봉사기에 의해서 읽는다.
4) /usr/dt/config/C/Xsession- 기정체계범위의 변수들은 X 봉사기로 읽지 못한다.
5) /usr/dt/config/C/Xconfig -기정체계범위변수들은 X 봉사기에 의해서 읽는다.
6) /usr/dt/bin/dtlogin - 표준기정변수들은 가장 낮은 우선권을 가진다.

자료형과 행동정의에 대하여 CDE 는 다음과 같은 우선권에 따라 .dt 파일들을 검사한다.

1) $HOME/.dt/types
2) /etc/dt/appconfig/types/C
3) /usr/dt/appconfig/types/C

자료형과 행동에 대해서는 나타난 첫 값이 리용된다는것을 기억하여야 한다. 만일 작업할 파일형 혹은 행동을 읽을수 없다면 그 파일의 앞에 있는 중복된 등록항목을 검사하거나 더 높은 우선권을 가진 파일에서 항목을 검사하여야 한다. CDE 가 파일형과 행동정의정보를 찾을수 있는 등록부들에 추가하기 위하여 환경변수 DTDATABASESEAR CHPATH 는 /etc/dt/config/Xsession 혹은 $HOME/.dtprofile 으로 설정될수 있다.

형태와 행동의 규정

구성파일들에서 형태와 행동자원들을 규정하려면 두가지 방법을 알아야 한다. 첫째로 자원을 규정하고 그 정확한 값을 주는것이며 둘째로 정확한 구성파일에서 자원들과 값을 규정하는것이다.

이 두가지 문제는 색과 서체들에서 제기된다. CDE 양식관리자는 색과 서체를 변경시키는데 도형대면부를 리용한다. 만일 응용프로그람이 직접 색 혹은 서체를 규정한다면 응용프로그람의 자원을 관리하는 양식관리자의 기능이 무시된다.

색 혹은 서체를 규정하는 대표적방법에는 다음과 같은것들이 있다.

- 시동지령행에서 선택항목을 규정한다.
- 응용프로그람의 app_defaults 파일에서 규정한다.
- 자원자료기지에 응용프로그람의 자원을 추가하기 위하여 Xrdb 보조응용프로그람을 리용한다.

CDE 가 시동될 때의 사건렬

다음 항목에서는 사용자가 CDE 에 가입할 때 진행되는 프로세스를 한걸음씩 고찰한다. 여기서 디스크클라스터처럼 분배된 환경을 가정하자. 이 프로세스는 하브체계의 기동으로부터 시작된다. (걸음 1) 걸음 4 까지는 X 봉사기들이 수행되는 프로세스와 가입화면이 현시되는 프로세스를 보여 준다. 걸음 6 까지는 사용자가 가입하는 프로세스를 보여 준다. 걸음 11 까지는 가입기간관리자가 사용자의 가입기간을 재창조하는 프로세스를 보여 준다.

1) 실행파일 dtlogin 은 **하브**(hub)기계와 매 클라스터의 정점에서 체계가 기동될 때에 발생하는 init 프로세스의 일부분으로서 시동된다.

2) dtlogin 은 가입프로쎄스를 조성하기 위한 자원목록을 얻기 위하여 /usr/dt/config/Xconfig 를 읽는다. 여기서 dtlogin 은 Xaccess, Xservers, Xresources, Xstartup, Xsession, Xreset 와 같은 파일들을 처음으로 알게 되며 많은 형태와 행동자원의 값들을 얻게 된다.
3) dtlogin 은 /usr/dt/config 에서 두 파일들을 읽는다.
    a. Xservers 혹은 Xconfig 에 있는 dtlogin*servers 자원설정에 의하여 식별되는 파일
    b. Xresources 혹은 Xconfig 에 있는 dtlogin*resources 자원설정에 의하여 식별되는 파일
4) dtlogin 은 X 봉사기와 매 국부적현시를 위한 후손 dtlogin 을 시동시킨다.
5) 매 후손 dtlogin 은 가입화면 dtgreet 를 표시한다.
6) 가입자와 통과암호가 유효하면 후손 dtlogin 은 일정한 환경변수들을 기정값으로 설정한다.
7) 후손 dtlogin 은 /usr/dt/config/Xstartup 을 수행시킨다.
8) 후손 dtlogin 은 /usr/dt/config/Xseeeion 을 수행시킨다.
9) Xsession 은 dthello 를 수행하고 판권화면을 표시한다.
10) Xsession 은 $HOME/.dtprofile 을 읽고 추가적인 환경변수들을 설정하거나 dtlogin 에 의하여 이미 설정된 값들을 무시한다.
11) 후손 dtlogin 은 가입기간관리자 dtseeeion 을 수행시킨다.
12) dtsession 은 적당한 가입기간을 재기억시킨다. 실례로 현재의 가입기간을 재기억하기 위하여 dtsession 은 $HOME/.dt/sessions/current 에 있는 dt.resources 와 dt.session 을 읽는다.

가입탈퇴는 반대순서로 진행된다. 이때 가입기간은 보관되며 dtlogin 은 /usr/config /Xreset 를 수행시킨다. Xreset 가 끝난 후에 dtlogin 은 걸음 4 의 가입화면을 다시 현시한다.

CDE 와 수행

CDE 는 한개의 응용프로그람만이 아니라 조작체계, X 창문체계, Motif 우에 있는 계층화된 성분들의 모임이다. CDE 혹은 의뢰기들이 시동되기전에 매개 층에서는 RAM 을 공유하게 된다. 이렇게 낮은 준위에서 리용되는 RAM 은 디스크와의 교환프로쎄스를 복잡하게 한다.

일부 경우에는 조작체계의 비용과 사용자응용프로그람의 요구로부터 도형사용자대면부에서 리용될수 있는 RAM 의 용량은 제한된다. 이러한 도형사용자대면부에서 리용되는 RAM 의 용량은 Motif 와 같은 창문관리자의 수행에 요구되는 량보다는 더 작다. CDE 작업공간관리자와 Motif 창문관리자는 거의 같은 RAM 의 용량을 요구하기때문에 사용자는 본질적으로 Motif 창문관리자수행을 위한 추가적인 RAM 이 없이도 여러개의 CDE 에 작업공간의 값으로서 추가된 도형환경을 리용할수 있다.

더 좋은 성능을 위한 전법

체계의 능력이 RAM 에 적재되여 있지 않다면 모든 사용자들은 수행방법을 개척하는데 일정한 시간을 소비하여야 할것이다. 만일 수행할 과제를 종모양의 곡선을 그리는데 비교한다면 기본문제는 테두리를 그리는것과 같다고 볼수 있다. 여기서 수행전법은 사용자가 테두리를 얼마나 크게 하는가 하는 문제이다.

가장 론리적인 방법은 작은것으로부터 시작하여 점차적으로 확대시켜 나가는것이다. 다시말하여 망우의 모든 체계들에서 최소한의 사용자환경으로부터 시작하는것이다. 그리고 점차적으로 사용자들이 성능저하를 알게 될 때 쏘프트웨어의 성분들을 추가하기도 하고 제거하기도 한다. 이러한 방법론은 몇주일이상 걸릴수 있는데 성분들을 추가하거나 며칠동안 작업하면서 체계성능변화의 효과와 실패준위를 결정하게 될 때 평가된다.

RAM 에서 CDE 와 관련한 가장 중요한 부분은 작업공간관리자, 가입기간관리자, 파일관리자들이다. 그중에서도 작업공간관리자는 RAM 에 상주(창문들을 이동시키고 작업공간을 절환할수 있다고 가정한다면)하고 있으므로 보다 더 중요한 부분으로 된다. CDE 작업공간관리자는 Motif 창문관리자보다는 중요하지 않다. 그러나 GUI 를 요구한다면 CDE 작업공간관리자는 반드시 필요하다. 가입기간관리자는 가입기간들을 보관하고 복구하기때문에 가입하고 탈퇴할 때에만 필요하다. 나머지 시간에 가입기간관리자는 작업을 하지 않고 RAM 의 밖으로 교환되게 된다. 현재작업가입기간의 보관은 하루작업을 끝낼 때 하는것이 좋지만 가입과 가입탈퇴의 성능을 개선하려면 보관하는것을 고려할 필요도 있다. 파일관리자는 주기적으로 작업하고 파일체계상태를 검사하기 위하여 RAM 에로 이행하며 파일관리자창문을 갱신하므로 비용이 많이 들수 있다. RAM 에 이행할 때 리용하던 탁상화면의 프로그람은 보관되며 나가게 된다.

아래에서 유용하게 리용될수 있는 일부 다른 방법들을 보여 준다.

Terminal Emulators	Xterms 는 dtterms 보다 더 작은 RAM 을 리용한다. dtterm 의 블로크방식기능을 필요로 하지 않는다면 말단을 모방하기 위하여 Xterm 을 선택하는것이 더 좋다.
Automatic Saves	일부 응용프로그람들은 주기적시간구간마다 자료를 자동적으로 보관한다. 비록 이 특징은 유익할수 있지만 수행의 관점에서 그 효과를 평가하여야 한다. 만일 응용프로그람의 사용자가 작업에 집중한다면 좋지만 그렇지 않다면 자동적보관의 특징을 리용하지 않는것이 좋다.
Scroll Biffers	말단모방자에서 큰 흐름완충기들은 실지로 편리할수 있으나 그것은 많은 RAM 을 소비한다. 비록 적당하게 큰 흐름완충기라고 해도 3 개 혹은 4 개 말단모방자들로 다중화될 때 많

은 RAM 을 소비하게 된다.

Backgorund Bitmaps	큰 비트매프의 리용은 피해야 한다. 그것들은 봉사기 X 의 크기를 증가시킨다. 가입기간안에서 흔히 큰 비트매프로의 절환을 피해야 한다. 만일 새로운 변경을 하려고 한다면 한개의 비트매프를 찾고 보통 파일 sessionetc 에 그것을 포함시킨 후 봉사기 X 를 시동하도록 하여야 한다. 비트매프를 배경에서 리용하려면 쪼각으로 그 비트매프를 반복시키는것이 좋다.
Front Panel	사용자들이 필요한것만큼 단추의 개수를 최소화하여 정면판을 재모형화하여야 한다. 이 전법은 RAM 에서 작업공간관리자크기를 감소시키고 가입과 가입탈퇴를 빨리 하게 한다.
Pathnames	가능한만큼 비트매프를 규정하는데 절대적경로이름을 리용하여야 한다. 비록 이것은 체계의 융통성을 감소시키지만 접근시간은 줄어 든다.

결 론

기정 CDE 는 리용에 편리하며 그 위력과 융통성이 주어 져 있지만 사용자는 자기의 요구에 맞게 작업환경에서 CDE 환경을 개별화하고 수행을 합리화할수 있다. 이때 필요한것, 그것을 변경시킬 순서, 그것을 어디서 정확히 변경시킬것인가에 대한 좋은 착상은 개발에 필요한 시간을 절약하게 한다. 이렇게 하는데서 CDE 의 모든 위력과 융통성은 사용자에게 항상 제공되여 있다.

제１６장. 망

망을 설치하는 조건에 따라 그 기능은 크게 변할수 있다. 체계에 접속된 ASCII 말단들만 가지는 고도로 집중되고 고립된 체계와 같은 조건에서 설치한다면 체계관리자는 망의 극히 작은 부분에만 관심하면서 설치할수 있다. 수천개의 체계들의 위상적지점들을 련결시키면서 망에 접속되는 널리 분산된 환경과 같은 다른 조건에서 설치한다면 체계관리자는 망에 상당히 많은 주의를 돌려야 한다. 이 두번째 방식에서 체계관리자가 망에 바치는 총 시간은 다른 모든 체계조직기능을 결합하는데 바치는 총 시간을 초과할수도 있다. 체계의 전반적관점에서 볼 때 첫번째 체계관리자는 망작업전반에 대하여 관심하지 않아도 되지만 반면에 두번째 체계관리자는 망작업전반에 대하여 관심하지 않으면 안된다. 그러므로 이 장에서는 대부분 UNIX 체계에서 적용될 망에 대한 내용을 포함시키기로 한다. 이 내용은 새로운 UNIX 체계의 변종으로 설치하여 본 실지 작업경험에 기초하고 있다. 만일 이 장에서 포함되여 있는 내용보다 더 많은 내용이 요구된다면 망정보의 좋은 원천으로 되는 다음의 책을 참고하기 바란다.

Bruce H. Huuter, Karen Brodford Huuter, UNIX Networks.(prentice Holl, ISDN 0-13-08987-1)

이 장에서 **설치**(Setup)가 주로 체계관리자들에 의하여 수행되기때문에 설치통보는 많은 망설치화면통보보다 토대적인 배경으로 제공된다. 여기서 취급하는 대부분 내용은 때때로 "인터네트봉사"라고도 부른다. 이 장에서는 망에 대한 일반적인 기초내용을 제시하려고 한다. 이것은 ARPA 와 버클리의 봉사들을 포함한다. 아래에서는 취급된 표제목록을 제시한다.

- 일반 UNIX 망배경
- 인터네트규약(IP)의 주소화(클라스 A, B, C)
- 부분망마스크
- ARPA services
- Berkerly 지령들
- 호스트이름의 대응
- 망파일체계(NFS)의 배경
- UNIX 망지령들

이 장에서는 많은 UNIX 지령들의 개괄과 실례들을 제공한다. 더 상세한 내용은 이 지령들에 대한 편람들에서 찾아보기 바란다. 여기에서 취급하는 많은 지령들에 대해서는 이 장의 마감에 있는 지령안내들을 참고하면 된다.

이 장에서 서술한 실례들에서는 Solaris, AIX 와 HP-UX 를 포함하여 체계의 변종들을 리용하고 있다.

UNIX 망

다른 콤퓨터와의 connecting 은 매 UNIX 망의 중요한 부분이다. 이것은 UNIX 를 리용하지 않는 기계를 포함하여 다른 UNIX 기계와의 접속을 의미한다. 기계들은 다른 체계에 가입하고 파일전송과 같은 과제를 수행할수 있도록 다른 기계에 기능적으로 접속되는 것과 함께 다른 기계와 물리적으로도 접속되여야 한다. 체계들사이의 파일전송과 가입기술을 제공하기 위하여 UNIX 체계에 많은 지령들이 존재한다. 이미 telnet 와 ftp 와 같은 ARPA 의 지령들도 알려 져 있다.

telnet 지령은 종류가 다른 환경에서 원격적가입을 허용한다. 실례로 UNIX 체계로부터 UNIX 가 아닌 체계에 telnet 를 줄수 있고 가입할수 있다. 원격체계에 가입한후에는 그 체계에서 수행하는 조작체계의 리해를 가지는것이 필요하다. 만일 체계들사이에 파일을 전송할 목적으로 다른 콤퓨터에 접속하려고 한다면 ftp 를 리용하여야 한다. 이 지령은 원격체계에서 수행하는 조작체계에 대한 리해가 없이도 임의의 두 체계사이에서 파일전송을 할수 있게 한다.

이 지령들을 UNIX 체계에서 제시되는 지령들과 얼마간 본질적인 측면에서 비교하기로 한다. UNIX 체계는 networking 에 대한 기능을 자기 체계에 이미전에 첨가하려고 하였다. ftp 와 telnet 지령들은 사용자 UNIX 체계와 다른 UNIX 체계들사이에 통신을 보장할수 있도록 더 개선된 지령들과 기능들을 가지고 사용자 UNIX 체계에서 사용된다. Berkely 지령이라고 알려 진 더 개선된 이 지령들은 파일복사와 등록부복사, 가입과 같은 많은 원격지령들을 수행하게 한다. 망에서 임의의 체계에 적재된 파일들과 작업하려는 지향은 끝없이 계속되고 있으며 이러한 요구들은 망파일체계(NFS)에 의하여 명백히 실현된다.

여기서는 UNIX 망에 대한 일부 기초지식을 보기로 한다.

IEEE802.3, TCP/IP 의 개괄

UNIX 체계작업에서 망에 대하여 리해하기 위해서는 우선 UNIX 체계에 존재하는 망의 성분들을 리해하는것이 필요하다. UNIX 체계에는 그림 16-1 에서 보여 준것처럼 망기능의 7 개층이 존재한다. 망작업처리에서 매 부분의 기능을 알도록 하기 위하여 아래의 4 개층은 미약하게 취급하는데 그것과 관련된 구체적내용들은 UNIX 체계에서 망을 구성하고 고장수리를 할 때 더 제시하기로 한다. 웃층들이 사용자와 기능적으로 밀접히 관계되기때문에 대다수 UNIX 체계관리자들은 망작업의 많은 시간을 여기서 보낸다. 밑의 층들은 그러나 이전 수준에서 리해하는것은 중요하며 그래서 체계의 전반적수행이 중요한 효과를 가지는 체계의 망작업의 수행을 개선하는데서 필요한 임의의 모형화를 수행할수 있도록 한다.

층번호	층이름	자료형식	설 명
7	응용층		사용자응용프로그람
6	표현층		준비된 응용프로그람
5	대화층		준비된 응용프로그람
4	전송층	파케트	TCP 로 조종되는 포구번호전송
3	망층	데타그람	IP 는 목적지 혹은 기정경로선택기로 직접 가는 경로조종
2	련결층	프레임	원천지주소와 목적지주소를 가지고 Ethernnet 혹은 IEEE 802.3 에서 통합된 자료
1	물리층		체계들사이에 물리적접속. 보통 동축케블 혹은 꼬임쌍선

그림 16-1. ISO/OSI 망층 기능들

그림 16-1 에서는 1 층으로부터 시작하여 아래 4 개 층에 대하여 서술한다. 이것은 국제적표준화조직인 열린체계의 호상접속(ISO/OSI) 모형이다. 이 층들은 망층들가운데서 관심 있는 층들을 보는데 도움이 될것이다.

물리층

시작층은 망작업에서 체계들사이의 호상접속이다. 물리층이 없이는 체계들사이에 통신할수 없고 실현하려고 하는 많은 기능들을 실현할수 없다. 물리층은 송신할 자료를 련결선을 통하여 이동될 상사신호들로 변경시킨다. (물리층은 언제나 리용될 련결선을 가진다고 가정한다.) 망대면부에로 이동한 정보는 련결선을 벗어나 다음 층에서 사용하도록 준비한다.

련결층

자기의 체계에 다른 국부적인 체계를 접속시키기 위해서는 국부적토막에 있는 다른 모든 체계와의 접속을 실현할수 있는 련결층을 리용하여야 한다. 이것은 IEEE 802.3 혹은 Ethernet 를 가지는 층이다. UNIX 체계는 이 통합화방법으로 둘다 봉사한다. 그 리유는 자료를 이 두 형식(IEEE 802.3 혹은 Ethernet)중의 한 형식으로 출구할수 있기때문에 통합화라고 부르는것이다. 자료는 프레임(자료의 다른 이름)으로 련결층에 전송되는데 이 프레임은 원천지주소 및 목적지주소 그리고 일부 다른 정보들을 덧붙여 전송된다. 두개의 다른 자료통합방법이 존재하기때문에 그것들이 많이 차이날것이라고 생각할수 있다. 그러나 이 추측은 옳지 않다. IEEE 802.3 와 Ethernet 는 거의 동일하다. 이런 리유로부터 많은 UNIX 체계들은 통합화의 두 형태들을 다 조종할수 있다. 그래서 사용자는 밑의 두 층에서 체계들사이에 물리적접속을 가지며 원천지주소와 목적지주소를 덧붙인 두 형식중

의 한 형식으로 통합된 자료를 가지게 된다. 그림 16-2 는 Ethernet 통합의 성분들을 목록화하고 IEEE 802.3 통합에 대하여 간단히 설명한다.

목적지주소	6byte	주소자료를 목적지에 보낸다.
원천지주소	6 byte	원천지로부터 주소자료를 보낸다.
형	2 byte	이것을 802.3 에서 길이계수기이다.
자료	46-1500 byte	802.3 에서 38-1492byte 이다 : 이 두 자료크기(MTU)에서 차이는 ifconfig 지령에서 볼수 있다.
검열합(crc)	4 byte	오유발견을 위한 검열합

그림 16-2. Ethernet 통합화

주의해야 할 중요한것은 IEEE 802.3 과 Ethernet 사이에 최대자료크기에서 1492byte 와 1500byte 로서 차이난다는것이다. 이것은 최대전송단위(MTU)이다. ifconfig 지령은 사용자대면부의 MTU 를 간단히 표시한다. Ethernet 에서는 자료를 프레임이라고 부른다. (다음층에서는 자료의 재통합화를 IP 에서 **데타그람**(Datagram)이라고 부르며 두 수준에서 통합화를 TCP 의 파케트라고 부른다.)

Ethernet 와 IEEE 802.3 이 같은 물리적접속에서 수행할것이라고 추측하고 있으나 두 통합화방법들사이에는 실지 차이가 있다. UNIX 체계에서 통합화의 망대면부설정에 대해서는 많이 고찰하지 않기로 한다.

망층

다음으로는 망층인 세번째 층을 고찰하기로 한다. UNIX 체계에서 이 층은 인터네트규약(IP)이라는 다른 이름도 가진다. 이 층에서 자료는 데타그람으로써 전송된다. 이것은 망을 통하여 자료의 경로화를 조종하는 층이다. 경로화된 IP 로 읽은 자료는 어떤 형태의 오유와 때때로 충돌하며 그것은 인터네트조종통보규약(ICMP)에 대한 통보로 원천체계에 거꾸로 전달된다. 여기서는 일부 ICMP 통보에 대하여 간단히 보기로 한다. ifconfig 와 netstat 는 이 경로화를 모형화하는데서 공통으로 리용되는 두개의 UNIX 지령들이다.

Ethernet 프레임안에서 IP 를 리용한 정보는 토막화된 자료로 끝날 만큼 내부적으로 편리하지 못하다. 이것은 자료의 웃층에서 사용자가 작업하기 편리하도록 실지적으로 자료의 재통합화를 한다.

IP 는 단순한 방법으로 경로화를 조종한다. 만일 자료를 사용자체계에 직접 접속된 목적지에 보내자면 그 자료를 직접 그 체계에 보낸다. 다른 한편 목적지가 사용자체계에 직접 접속되지 않았으면 자료는 기정경로화를 통하여 보낸다. 기정경로자는 다음으로 그 목적지에서 자료를 받도록 하는 책임을 지닌다. 이 경로화는 간단히 취급하였기때문에 리해하기 힘들수 있다.

전송층

전송층은 망층에서 다음 수준이다. 이것은 포구들을 리용하여 통신한다. TCP 는 이 수준에서 볼수 있는 가장 일반적인 규약인데 그 형식들은 포구들사이에 보내는 파케트들이다. 프로그람에서 리용된 포구는 보통 (TCP 와 같은)규약과 함께 /etc/services 에서 정의된다. 이 포구들은 telnet, rlogin, ftp 와 같은 망프로그람에서 리용된다. 일람표를 해석하여 보면 이 층은 포구와 관련되는 제일 높은 층으로서 사용자는 여기서 프로그람을 볼수 있다.

인터네트규약(IP)주소화

인터네트규약주소(IP 주소)는 클라스 "A", "B", "C"주소중의 하나이다. ("D"와 "E"클라스의 주소도 있지만 여기서는 취급하지 않는다.) 클라스 "A"망은 클라스 "B"망 혹은 "C"망보다 망단위당 더 많은 매듭을 제공한다. IP 주소들은 4 개 마당들로 구성된다. IP 주소를 4 개 마당으로 쪼개는 목적은 매듭(혹은 호스트)주소와 망주소를 정의하기 위해서이다. 그림 16−3 은 클라스와 주소들사이의 관계를 보여 준다.

주소클라스	망	망단위당 매듭	망을 정의하는 비트수	망단위당 매듭을 정의하는 비트수
A	약간	대부분	8bit	24bit
B	많다.	많다.	16bit	16bit
C	대부분	약간	24bit	8bit
예정된	-	-	-	-

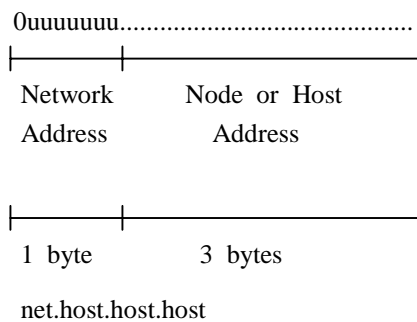
그림 16−3. 인터네트규약(IP)주소들의 비교

이 비트패턴들은 비트수가 매 클라스에서 망과 매듭의 범위를 정의하는데서 중요한 것이다. 실례로 클라스 "A"의 주소는 망을 정의하기 위하여 8bit 를 사용하고 클라스 "C"의 주소는 망을 정의하는데 24bit 를 사용한다. 그러므로 클라스 "A"의 주소는 클라스 "C"의 주소보다 더 작은 망을 지원한다. 그러나 클라스 "A"의 주소는 클라스 "C"의 주소보다 망단위당 더 많은 매듭을 제공한다. 주소지정방법의 세부들을 한 걸음 더 나아가서 보면 그림 16−4 와 같이 이 주소클라스들과 련관된 개별적인수들처럼 볼수 있다.

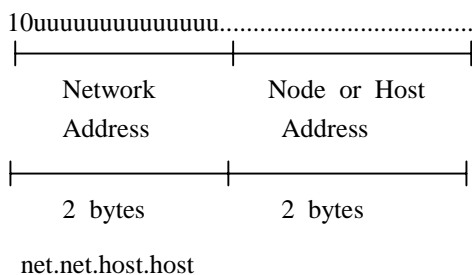
이 주소들은 /etc/hosts파일들이 서술될 때 취급하는 여러가지 설정파일들에서 사용된다. 사용자망에서 개별적인 대면부는 고유한 IP주소를 가져야 한다. 두개의 망대면부를 가지는 체계들은 두개의 개별적인 IP주소를 가져야 한다.

주소 클라스	봉사하는 망의 개수	망 단위당 매듭의 수	주 소 범 위		
A	127	16777215	0.0.0.1	-	127.255.255.254
B	16383	65535	128.0.0.1	-	191.255.255.254
C	2097157	255	192.0.0.1	-	223.255.254.254
예정된	-	-	224.0.0.0	-	255.255.255.255
2 진형식에서 32bit 주소를 찾으면 주소클라스를 어떻게 결정하는가 하는것을 알수 있다.					

클라스 "A"

net.host.host.host

클라스 "A"의 주소는 첫번째 비트설정을 0 으로 한다. 매개 망이 얼마나 많은 매듭들을 가지는가 하는것은 매듭 혹은 호스트주소에 배당된 모든 비트들로써 알수 있다.
클라스 "A"주소의 첫번째 비트는 0 이고 망부분의 나머지 7bit 들은 망을 정의하는데 사용된다. 그리고 나머지 3 Byte 는 한 망안의 매듭들을 정의하는데 사용된다.

클라스 "B"

net.net.host.host

클라스 "B"주소는 첫번째 비트를 1 로, 두번째 비트를 0 으로 설정한다. 여기서는 클라스 "A"주소보다 더 많은 망들을 제공하지만 개별적 망에서의 매듭들은 더 작게 제공한다. 클라스 "B"주소에서 2byte 는 망주소에 배당되고 2byte 는 매듭주소에 배당된다.

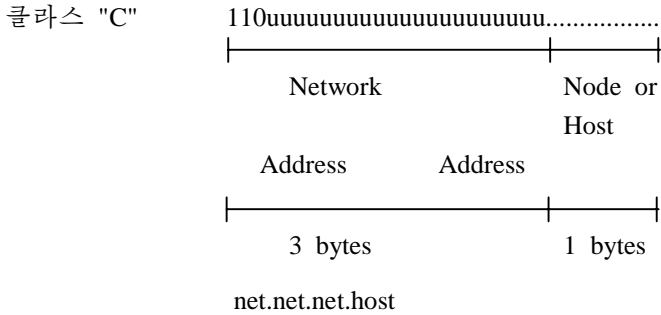
net.net.net.host

클라스 "C"의 주소는 첫번째 비트와 두번째 비트는 1 로, 세번째 비트는 0 으로 설정한다. 망들의 최대수와 개별적인 망에 련결할수 있는 최소매듭의 수는 클라스 "C"의 주소와 관계된다. 클라스 "C"의 주소에서 3Byte 는 망에 분배되고 1Byte 는 매 망에 접속할수 있는 매듭에 분배된다.

그림 16-4. 주소클라스

부분망마스크

사용자 UNIX 체계는 IP 데타그람이 그자신의 부분망의 호스트를 위한것인가, 같은 망이지만 다른 부분망의 호스트를 위한것인가, 다른 망의 호스트를 위한것인가를 결정하는 부분망마스크를 사용한다. 부분망들을 리용하여 사용자는 한개 부분망의 일부 호스트들과 다른 부분망의 또 다른 호스트들을 리용할수 있다. 부분망들은 경로선택기들이거나 부분망들을 련결하는 다른 망회로들에 의하여 구분될수 있다.

경로를 선택하기 위하여 사용자경로선택기가 사용하는 주소적인 측면들은 망과 부분망뿐이다. 부분망마스크는 주소의 호스트부분을 마스크한다. 주소의 부분이 호스트, 부분망, 망이라는것을 알 때에만 그러한 방법으로 망주소를 설정할수 있기때문에 사용자는 부분망마스크를 리용하여 체계를 호스트와 부분망에 대한 IP 주소의 비트들로써 인식할수 있게 한다.

가장 단순한 형태로 사용자가 부분망마스크화를 어떻게 하는가 하는것은 IP 주소의 어느 부분이 호스트를 정의하고 어느 부분이 망을 정의하는가를 규정한다. 보통 책들에서는 부분망마스크로 작업할 때 가장 혼돈할수 있는 경우가 그림 16-5 에서 보여 주는것이라고 지적하고 있다.

주소클라스	10 진수	16 진수
A	255.0.0.0	0xff000000
B	255.255.0.0	0xffff0000
C	255.255.255.0	0xffffff00

그림 16-5. 부분망마스크들

그러나 이러한 사고방식은 망에 가능한껏 많은 비트들을 배당하고 호스트에 가능한껏 많은 비트들을 배당하며 부분망이 리용되지 않는다는것을 가정하고 있다. 그림 16-6은 클라스 B 주소에서 부분망리용의 실례를 보여 준다.

주소클라스	클라스 B		
호스트 IP 주소	152.128.	12.	1
분 해	망	부분망	호스트 id
비트수	16bit	8bit	8bit
10 진으로 부분망마스크	255.255.	255.	0
16 진으로 부분망마스크	0xffffff00		
같은 부분망에서 다른 호스트실례	152.128.	12.	2
다른 부분망에서 호스트의 실례	152.128.	13.	1

그림 16-6. 클라스 B 의 IP 주소와 부분망마스크실례

그림 16-6 에서 부분망마스크의 첫 두 바이트(255.255)는 망을 정의하고 세번째 바이트(255)는 부분망을 정의하고 네번째 바이트(0)은 호스트 ID 에 배당된다. 비록 클라스 B 주소의 부분망마스크는 이미 본 기정부분망마스크의 모양으로 나타나지 않았지만 255.255.255.0 의 부분망마스크는 부분망이 놓이는 클라스 B 망에서 널리 리용되고 있다.

UNIX 체계는 부분망마스크 255.255.255.0 을 리용하여 152.128.12.1 과 152.128.13.1 이 다른 부분망이라는것을 결정하기 위하여 어떻게 비교를 수행하는가? 그림 16-7 은 이 비교에 대하여 보여 주고 있다.

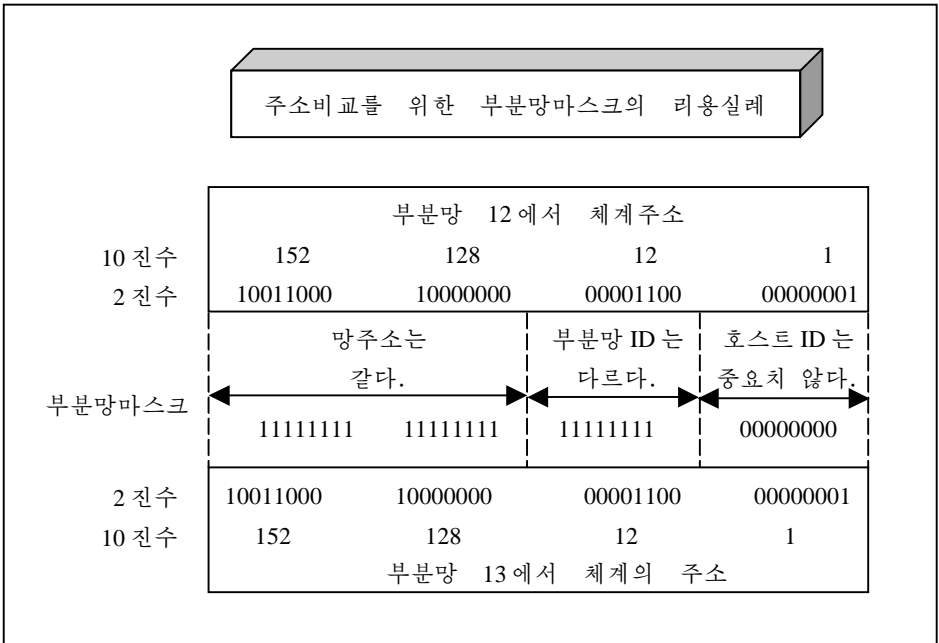



그림 16-7. 주소비교에 부분망마스크를 리용한 실례

그림 16-8 은 개별적인 부분망들에서 이 두 체계를 보여 주고 있다.

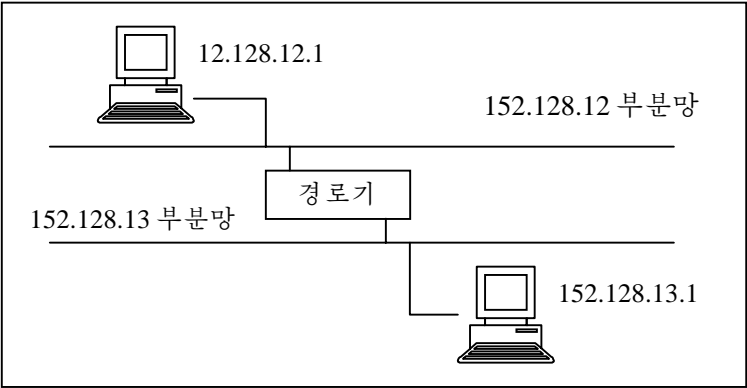



그림 16-8. 개별적인 부분망들에서 클라스 B 체계

망, 부분망, 호스트 ID 마당들을 서술하기 위하여 8bit 체계를 리용해야 한다는것은 아니다. 실례로 만일 호스트 ID 에 대하여 부분망마당의 일부분을 리용하려고 한다면 8bit 체계를 리용하여도 된다. 이렇게 취급하는것이 좋은 리유로 되는것은 앞으로 확장가능성을 고려한데 있다. 현재는 같은 부분망의 부분으로 될 부분망 12, 13, 14, 15 를 리용하려고하고 있지만 앞으로는 개별적부분망에서 이것을 택할수 있을것이다. 그림 16-9 는 이 설정에 대하여 보여 준다.

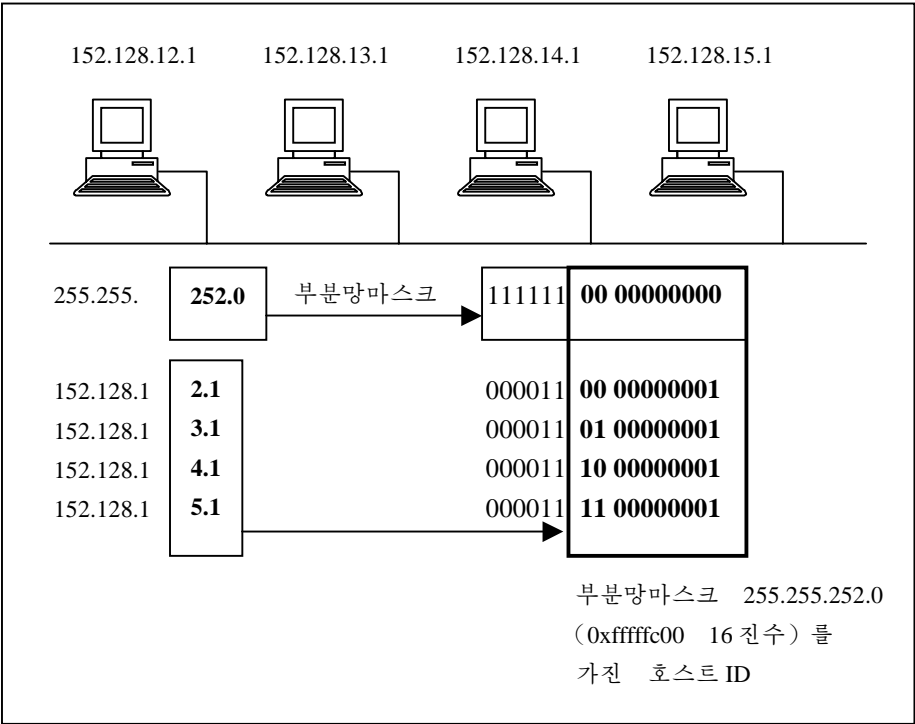



그림 16-9. 부분망마스크를 리용하는 앞으로의 확장성

이 체계들은 부분망과 표준적으로 관련된 세번째 바이트의 부분이 호스트 ID 에 리용되며 같은 부분에 접속된다. 앞으로 부분망마스크는 255.255.255.0 으로 교체될수 있다. 네개의 개별적부분망들은 12, 13, 14, 15 이다. 이것들을 개별적부분망에로 경로화하는 경로조종기를 리용할 때 요구될것이다.

이제 ISO/OSI 모형의 더 높은 수준으로 절환하고 일부 망의 기능들을 고찰하기로 한다.

망의 리용

ISO/OSI 모형은 망에서 층들의 호상관계를 시각적으로 보여 준다. 그러나 이 모형은 망을 리용하기 위하여 어떻게 해야 하는가를 의미하는것은 아니다. 우리의 체계에서 수행할수 있고 고찰할 가치가 있는 광범히 리용되고 있는 두개의 망봉사들로서는 ARPA 와 NFS 를 지적할수 있다.

우리가 고찰하는 체계에서 주목하는 첫번째의 망제품은 때때로 ARPA 봉사라고 부르기도 하지만 여기서는 간단히 ARPA 라고 표현한다. ARPA 는 ARPA 봉사기와 버클리의 봉사기들의 결합이다. ARPA 봉사기는 다른 조작체계를 수행하는 체계들을 통하여 통신을 지원하며 버클리의 봉사기는 UNIX 체계들을 지원한다. 다음의 항목들에서는 가장 공통적인 ARPA 와 버클리의 지령들을 해설한다. 비록 많은 프로그람들은 이 매개 봉사기들 안에서 수행할수 있지만 다음의 지령들은 UNIX 세계의 한 조작체계에서만 거의 공통적으로 리용된다. 일부 경우에 이 지령들이 어떻게 리용되는가를 보여 주는 실례들도 제시되여 있다. 이 대부분 실례들에서는 국부적호스트는 system1 이고 원격적호스트는 system2 로 되여 있다.

ARPA 봉사(각이한 OS 를 가진 체계들사이의 통신)

파일전송규약 (ftp)   한 체계로부터 다른 체계에로 파일 혹은 다중파일들을 전송한다. 이것은 흔히 UNIX 작업기와 windows PC, VAX 등 콤퓨터들사이에 파일전송에 리용된다. 다음의 실례는 system2(원격호스트)로부터 /tmlp/krsort.c 파일을 system1(국부호스트)의 국부등록부에 복사하는것을 보여준다.

	설 명
$ftp system2	ftp 지령주기
system2 에 접속된다.	
system2 의 FTP 봉사기(방안 4.1)는 준비된다.	
이름 : 뿌리(system2.root)	system2 에 가입
뿌리에 요구되는 통과암호	
password:	통과암호입력

```
사용자뿌리는 가입되였다.
원격체계의 형의 UNIX 이다.
파일전송에 2 진방식리용
ftp>cd/tmp                                  system2 에서 /tmp 로 cd 설정
cwd 지령성공
ftp>get  krsort.c                           krsort.c 파일읽기
PORT 지령성공
krsort.c 에 대한 BINARY 방식자료접속열기
전송완성
0.08 초에 전송된 2896byte
ftp>bye                                     ftp Exit
goodbye
$
```

이 실례에서 두 체계들은 UNIX 체계에서 수행되고 있으나 ftp 로 준 지령들은 조작체계에 독립적인것이다. 등록부변경을 위한 cd 와 get 지령들은 ftp 가 수행되는 임의의 조작체계에서 작업하는데도 리용된다. 만일 바로 ftp 지령과 어떤 류사한 지령을 준다면 종류가 다른 망환경에서도 통보전송이 어렵지 않다는것을 알수 있을것이다. 종류가 다른 환경에서도 UNIX 체계를 리용할수 있으며 한 체계로부터 다른 체계에로의 파일복사 및 등록부복사에 ftp 를 리용할수 있다. ftp 가 널리 리용되기때문에 더 일반적으로 리용될수 있는 ftp 지령들가운데서 일부 지령들만 해설하기로 한다.

ascii ASCII 로 전송될 파일의 형을 설정한다. 이것은 한 체계로부터 다른 체계에로 ASCII 파일을 전송한다는것을 의미한다. 이것은 보통 기정값이지만 그렇게 설정하지 않을수도 있다.

실례: ascii

binary 2 진으로 전송될 파일의 형을 규정한다. 이것은 한 체계로부터 다른 체계로 2 진파일을 전송한다는것을 의미한다. 실례로 만일 UNIX 를 리용하지 않는 체계에는 복사할 응용프로그람을 가지고 있고 UNIX 체계의 한 등록부에는 그 파일이 있도록 하자면 2 진전송을 해야 한다.

실례:binary

cd 원격호스트에서 규정된 등록부에로 작업등록부를 변경한다.

실례: cd /tmp

dir 원격체계의 등록부의 내용을 화면에 목록화하거나 만일 국부적체계의 파일이름이 규정된다면 원격체계의 등록부의 내용을 국부적체계의 파일에 목록화한다.

get 규정된 원격파일을 규정된 국부적파일에 복사한다. 만일 국부적파일이 규정되지 않으면 원격파일이름을 리용한다.

lcd 국부적호스트에서 규정된 등록부로서 변경한다.

실례: lcd /tmp

ls 원격체계의 등록부의 내용을 화면에 목록화하거나 만일 국부적파일이름이 규정된다면 국부적체계에서 파일에 목록화한다.

mget 원격호스트로부터 국부적호스트에로 여러 파일을 복사한다.

실례:mget *.c

put 규정된 국부파일을 규정된 원격파일에로 복사한다. 만일 원격파일이름이 규정되지 않았다면 국부파일이름을 리용한다.

실례: put test.c

mput 국부적호스트로부터 원격적호스트에로 여러 파일들을 복사한다.

실례: mput *.c

bye/quit 원격적호스트의 접속을 닫는다.

실례: bye

다른 ftp 지령들은 여기서 서술한것에 추가하여 리용할수 있다. 만일 지령에 대한 더 많은 정보가 요구되거나 추가적 ftp 지령들을 보려고 한다면 ftp 에 대한 UNIX 의 편람들을 보기 바란다.

telnet telnet 규약을 리용하여 다른 호스트와 통신하는데 리용한다. telnet 는 후에 서술될 rlogin 을 리용하여 변경할수 있다. 다음의 실례는 원격호스트인 system2 와 telnet 가 접속을 어떻게 설정하는가를 보여 준다.

	설 명
$ telnet system2	
Connected to system2.	system2에로 telnet
AIXversion 4 system2	
login:root	system2에 뿌리로 가입
password:	통과암호입력
Welcome to system2.-rs6000 aix4.3.1.0	
$	system2에서 AIX입력대기문

버클리지령(UNIX체계들사이의 통신)

Remote Copy(rcp)

이 프로그람은 한 UNIX체계로부터 다른 UNIX체계에로 파일들과 등록부들을 복사하는데 리용한다. system1로부터 /tmp/krsort.c를 system2에 복사하려면 다음의 지령을 주어야 한다.

$ **rcp system2:/tmp/krsort.c /tmp/krsort.c**

어떤 망을 구성하는데서는 이 기능의 수준에서 읽기 위하여 순차적으로 파일을 만들것을 요구한다. 이 실례에서 지령을 준 사용자는 두 체계들사이에서 지령이 동등하다고 간주하는데 rcp를 리용하여 한 체계로부터 다른 체계에 파일들을 복사한다. (이 항목들에 대하여서는 간단히 서술하였다.)

Remote login(rlogin)

원격 UNIX체계에로 가입을 제공한다. system1로부터 system2에로 원격가입을 위해서는 다음과 같이 하여야 한다.

$ rlogin **system2**
password:
Welcome to system2
$

만일 사용자가 rlogin지령을 줄 때 통과암호를 사용하였다면 사용자들이 두 체계에서 동등하지 않다는것을 의미한다. 만일 통과암호를 사용하지 않았다면 사용자들은 동등하다는것을 의미한다. 사용자는 지령 rlogin의 부분으로 되는 system과 user를 리용하여 rlogin system -1 user로 줄수 있다.

Remote shell(remsh) remsh지령으로 사용자는 한 UNIX체계에서 다른 UNIX체계에로 원격적으로 수행될 지령을 줄수도 있고 결과를 국부적으로 표시할수도 있다. 이 경우에 remsh는 /tmp/krsort.c에 대

한 긴 목록을 아래에서와 같이 보여 준다. 이 지령은 system2 에서 수행되지만 결과는 지령을 입구한 system1 에 표시된다.

```
$ remsh    system2   ll   /tmp/krsort.c
-rwxrwxvwx  |   root  sys  2896  sept   1   10:54  /tmp/krsort.c
$
```

이 경우에 system1 과 system2 의 사용자들은 같은 자격을 가져야 하지만 다른 경우에는 이 지령의 수행을 거절한다.

Remote who(rwho) 원격 UNIX 체계에 사용자들을 찾는다. 아래에서는 rwho 의 결과를 제시하고 있다.

```
$ rwho
root   system1:ttyu0           Sept  1 19:21
root   system2:console Sept  1 13:17
tomd system2:ttyp2             Sept  1 13:05
|        |         |                    |          |>가입시간
|        |         |                    |>가입날자
|        |         |>말단행
|          |기계이름|
|>사용자이름
```

rwho 로 작업을 위하여서는 rwho 의 데몬(rwhod)를 수행하여야 한다. 추가적으로 이것을 포함하는 다른 지령 r 들을 리용할수 있다. 또한 이 지령들은 UNIX 의 한 방안에서 다른 방안으로 갈 때 약간 다르게 나타나므로 여기서는 자기의 UNIX 체계에서와 정확히 같은 지령 r 를 수행시킬수는 없다.

호스트이름대응

망과 관련하여 결정해야 할 가장 중요한것은 ARPA 에서와 자기 체계에서 호스트이름대응이 어떻게 실현되는가 하는것이다. 호스트이름대응에는 세개의 기술이 리용될수 있다.

- Berkeley Internet Named Domain(BIND)
- Network Information Services(NIS)
- UNIXfile/etc/hosts

가장 일반적이고 가장 단순한 호스트이름대응의 실현방법은 /etc /hosts 로 하는것이다. 그러므로 다음 항목부터는 이 기술에 대하여 해설한다. 아마 사용자들은 NFS, ARPA 와 다른것을 포함하여 망과 관련된 많은 참고문헌들에서 UNIX 방안에 대한 좋은 망편람들이 있다는것을 기억하고 있을것이다. 만일 앞에서 지적된 내용보다 망에 대하여 더 많은

것을 알려면 이와 관련한 편람들을 참고하기 바란다.

앞에서 본것처럼 /etc/hosts 파일을 리용해보면 개발되여 있는 많은 체계들의 환경과 대단히 차이난다는것을 알수 있다. 이 해결을 위하여 다른 모든 체계들에 보급되여 있고 날자로 유지되게 하는 한개의 /etc/hosts 파일이 출현하게 되였다.

령역이름체계(DNS)는 큰 환경에서 널리 리용된다. DNS 는 이름들을 주소로 다시 풀기 위하여 버클리의 인터네트이름령역(BIND)봉사를 리용한다. 이름자료에 대한 요청을 만족시키는 이름봉사기들이 있다. 이것은 봉사기측이 BIND 이라는것이다. 의뢰기측이 BIND 에 있고 한편 이름을 해결하기 위하여 이름봉사기에 입장하는 해결자라고 부르는 BIND 가 있다. 의뢰기-봉사기모형을 리용할 때 매 체계에 대치되는 장소들이 적게 필요하기때문에 BIND 는 이름을 만드는 통보를 유지하는데서 아주 쉽게 한다.

의뢰자들은 해결자를 모형화하기 위하여 /etc/resolv.conf 라는 파일을 리용한다. 이름봉사기와 그것에 대응하는 주소들은 해결하는 정보의 열쇠들로 된다.

이 해결은 큰 환경에서 체계이름들과 주소들의 유지를 더 쉽게 한다. 체계조직은 DNS 와 BIND 을 설치할 때 이것들에 대하여 먼저 실험해 보아야 한다. 사용자의 관점에서 볼 때 그 실험에 대해서 많이 알 필요는 없을것이다. 앞으로는 체계항목들에 초점을 집중할 대신에 사용자들이 더 관심을 가지는 일부 프로그람들에 주목할것이다. 프로그람들에 대한 더 깊은 의미를 알고 리용되도록 하기 위한 수단들을 일부 배경적으로 제시할것이다. 일반적으로 체계관리의 관점에서가 아니라 사용자의 관점에서 망과 관련된 내용들을 고찰하겠다.

/etc/hosts

이 파일은 접속된 다른 체계에 대한 정보를 포함한다. 이 파일은 매 체계의 인터네트주소, 체계이름, 체계이름에 대한 임의의 별명들을 포함한다. 만일 /etc/hosts 파일이 망에서 체계의 이름들을 포함하도록 변경된다면 이 파일은 다른 체계에 rlogin 을 위한 기초로 제공될것이다. 비록 다른 UNIX 체계에서 이제 지령 rlogin 을 사용할수 있다면 사용자는 다른 체계에서 지령 rcp 혹은 지령 remsh 를 사용할수 없을것이다. 비록 remsh 와 rcp 의 기능을 추가하기는 쉽지만 모든 체계들에 항상 설치되지 않으므로 안전을 확인해야 한다. 여기서 /etc/hosts 파일의 실례를 준다.

이 파일은 다음의 형식을 가진다.

```
127.0.0.1        localhost     loopback
15.32.199.42     a44120827
15.32.199.28     a4410tu8
15.32.199.7      a4410922
15.32.199.21     a4410tu1
15.32.199.22     a4410tu2
15.32.199.62     a4410730
15.32.199.63     hpxterm1
```

15.32.199.64 a4410rd1
15.32.199.62 a4410750 hp1

이 파일은 다음과 같은 형식을 가진다.

< internet_주소> <공식적호스트이름> <별명>

인터네트규약주소(IP 주소)는 클라스 A, B, C 의 주소이다. 클라스 "A"의 망은 클라스 "B" 혹은 "C"의 망보다 매 망단위당 더 많은 매듭을 제공한다. 4 개의 마당으로 IP 주소를 분해하는 목적은 매듭(혹은 호스트)주소와 망주소를 정의하기 위한것이다. 그림 16−3 부터 그림 16−6 까지에서는 이 클라스들에 대하여 상세히 서술하였다.

우의 /etc/hosts 파일은 클라스 "C"의 주소를 포함한다고 가정하고 있으며 가장 오른쪽 마당은 호스트 혹은 매듭의 주소, 다른 3 개의 마당들은 망주소를 함축하고 있다.

이미 서술된 ARPA 혹은 버클리의 지령들중 하나를 사용할 때 /etc/hosts 로부터 공식적호스트이름 혹은 별명을 리용한다. 실례는 다음과 같은 ARPA 지령의 작업에 보여 준다.

$ telnet a4410750 혹은 $ telnet hp1

류사하게 다음과 같은 버클리의 지령들의 작업을 보여 준다.

$ rlogin a4410750 혹은 $rlogin hp1

/etc/hosts.equiv

체계는 사용자가 원격체계에 rlogin 할 때 통과암호의 요구를 하지 않도록 설치할수 있고 또한 이 파일편집에 동등한 호스트로 설정할수도 있다. 이미 언급한것처럼 이것은 항상 사용되지 않는것으로서 안전을 파괴하는것을 고려한 기술이다. 가입이름들이 통과암호로 통과하기 위하여서는 /etc/hosts.equiv 의 원격체계와 국부체계에서 같아야 한다. /etc/hosts.e-quiv 에서는 모든 동등한 호스트들의 목록 혹은 동등하게 하려고 하는 호스트와 사용자이름목록을 목록화할수 있다. 이제부터는 사용자들이 이 체계에서 동등한 사용자들로 되기때문에 rep 와 remsh 를 리용할수 있다. 여기서는 일반적으로 망에 모든 호스트이름들을 입구할수 있다. 아래에서는 /etc/hosts.equiv 의 리용실례를 보여 주고 있다.

a4410730
a4410tu1
a4410tu2
hpxterm1

a4410827
a4410750

사용자들은 /etc/hosts.equiv 의 리용의 잠재적인 안전위험에 대하여 기억하고 있어야 한다.

만일 사용자가 통과암호없이 원격체계에 가입할수 있다면 망에서 전반적인 안전수준은 감소될것이다. 비록 사용자들이 원격체계에 가입할 때 통과암호를 입구하지 않도록 편의를 제공한다고 해도 /etc/hosts.equiv 에서 제시된 매 사용자는 망에 접근할수 있다. 만일 모든 체계에서 모든 파일들과 등록부들에 모든 허용성에 대한 설정이 담보되였다면 어떤 체계에 누가 입장하는지 알수 없다. 그러나 실지로 UNIX 세계에서의 허락은 그들이 무엇을 봉사받는지 모르게 된다. 사용자는 다른 사용자들이 실지로 입장해서는 안될 파일을 복사하려는 시도에 대하여 엄격히 통제하기 위한 수단을 가지고 있어야 한다.

/.rhosts

이 파일은 상급사용자를 위한 /etc/hosts.equiv 이다. 만일 사용자가 뿌리로서 가입한다면 사용자는 /etc/hosts.equiv 와 꼭 같은 정보로 모형화하는 파일을 요구한다. 그러나 만일 그렇게 한다면 임의의 체계에서 상급사용자에게 뿌리통과암호없이 원격체계에 가입한 망의 안전위험을 증가시킬것이다. 만일 망에 확실한 자료가 있고 안전성이 담보됨이 없이 자료를 유지할 100%확신이 있는 조건에서 상급사용자들만이 체계에 원격적으로 가입할수 있다면 통과암호가 필요없이 /.rhosts 를 설정할수 있다. 그러나 안전성의 관점에서는 이런 설정을 찬성하지 않을것이라는것은 뻔하다.

만일 적당한 변경이 /etc/hosts, /etc/hosts.equiv, /.rhosts 에서 적당한 등록항목에 만들어 진다면 사용자는 버클리의 지령들 rcp, rlogin, remsh, rwho 를 포함하여 ARPA 봉사의 지령들 ftp, telnet 를 리용할수 있다.

여기서는 가장 공통적으로 리용되게 되는 ARPA 봉사의 설정과 수행을 위한 적당한 파일들의 설치과정에 대하여 서술하게 된다. 때로는 DNS/BIND 와 같은 더 개선된 기능이 요구될수도 있다. 사용자체계는 DNS/BIND 를 가지거나 이 항목의 전반에서 포괄하는 일부 혹은 모든 지령들을 사용할수 있게 하는것과 같은 류사한 기능들도 설정될수 있다.

망파일체계(NFS)

NFS 는 사용자체계의 국부적인 체계인것처럼 원격체계에 디스크를 제공하게 한다. 류사하게 NFS 가 원격체계에서 국부적인것처럼 보이도록 그 원격체계는 사용자의 국부적 디스크를 제공한다. 이 기능을 실현하기 위한 NFS 의 모형화는 단순하다.

아래에서는 NFS 의 모형화를 위한 걸음들을 서술한다.

1) NFS 를 시작한다.

2) 사용자체계는 NFS 의뢰기의 기능을 수행하거나 NFS 봉사기기능을 수행하거나 혹은 이 두가지 기능을 수행하도록 규정한다.
3) 원격체계로 제공될수 있는 사용자국부적파일체계를 규정한다.
4) 사용자는 원격디스크들을 자기체계에서 국부적인것처럼 제공하고 보도록 규정한다.

ARPA 로서는 NFS 에 대하여 다른 관점도 가질수 있지만 최근에 매번 UNIX 설치에 리용한 NFS 기능을 알고 있는 정도로 다시 한번 설명한다.

NFS 는 많은 사용자들의 요구를 만족시키도록 체계에 설치되기때문에 NFS 와 관련된 기능을 알고 있어야 한다. 아래에서는 일반적으로 리용되는 NFS 항목들을 설명한다.

Node	콤퓨터망에 덧붙거나 혹은 콤퓨터망의 부분인 콤퓨터체계이다.
Client	다른 매듭(봉사기)으로부터 자료 혹은 봉사를 요청하는 매듭이다.
Server	망의 다른 매듭(의뢰기)에 자료 혹은 봉사를 제공하는 매듭이다.
File System	디스크구획 혹은 론리적인 권이다.
Export	NFS 를 리용하여 원격매듭을 제공하도록 파일체계를 취한다.
Mount	NFS 를 리용하여 원격파일체계에 입장한다.
Mount Point	NFS 파일체계가 설치된 체계에서의 등록부이름이다.
Import	원격파일체계를 설치한다.

규정된 모형의 일부 과제들과 관계되는 파일들은 UNIX 방안마다 다르다. 아래에서는 NFS 모형과 관련된 일부 일반적과제들과 실레들을 제시한다. 물론 체계관리자는 UNIX 방안의 구성에 대하여 상세히 알고 있어야 한다.

사용자체계는 NFS 의뢰기기능을 수행하거나 NFS 봉사기기능을 수행하거나 혹은 두 기능을 다 수행하여야 한다. NFS 의 봉사를 위하여 수행되여야 할 데몬들도 있다. 이 두 과제들은 UNIX 방안에서 일정한 차이가 있다.

사용자체계는 다음 국부적접근을 할 원격파일체계를 받아들이기도 하고 다른 체계에 의하여 입장될 국부적파일체계를 내보내기도 한다.

국부적으로 설치될 원격파일체계는 /etc/fstab, /etc/vfstab, /etc/filesystems 들중의 어느 하나와 류사한 등록항목을 가지지만 언제나 파일은 다음과 같이 파일체계를 설치하기 위하여 리용된다.

```
system2:/opt/app3  /opt/app3  nfs  rw,   suid  0  0
```

이 경우에는 /opt/app3 과 같이 system2 에서 /opt/app3 을 국부적으로 설치하고 있다. 이것은 앞에서 보여 준 NFS 의 설치법이다.

국부적파일체계가 설치된 모든 원격체계(의뢰기)들을 보기 위하여 지령 showmount 을 리용할수 있다. 이 지령은 대부분의 UNIX 방안들에서 봉사되고 있다. showmount 는

NFS 를 가진 의뢰기에 의해서 자주 설치되는 파일체계를 결정하는데 사용된다. showmount 의 출력은 호스트이름과 의뢰기에 의해서 설치된 등록부들을 목록화하므로 특별히 읽기 쉽게 되여 있다. showmount 지령에는 다음과 같은 3 가지 선택항목들을 리용할수 있다.

- -a "이름:등록부"형식으로 출구한다.
- -d 의뢰기에 의하여 원격적으로 설치된 모든 국부적등록부들을 목록화한다.
- -e 보내는 파일체계들의 목록을 출구한다.

다른 망지령들과 설치

망의 설치과정은 두개의 망과 체계관리자에 관한 집중적인 계획화의 실험과정이다. 두개의 망환경들은 동등한것이 아니다. 거기에는 사용자체계를 접속하기 위한 특징적인 전자요소들이 많다. 또한 거기에는 다른 체계와 망을 구성하기 위한 접속검열과 관련된, 필요한 많은 지령들이 있다. 어떤 문제가 제기된다면 사용자는 그것들을 처리할수 있도록 일부 망지령들을 알고 있어야 한다. 만일 UNIX 체계가 경로조종기, 관문, 다리와 같은 대면부를 가져야 하는 일부 망의 하드웨어와 작업한다면 망을 설치하는 과정에서 어떤 까다로운 문제와 만날수 있다는것도 보충적으로 강조해 둔다. 이런 경우에 경로조종기와 UNIX 체계를 접속시키는 한가지 실례를 준다. 동시에 아주 편리한 일부 망지령들을 제시한다.

그림 16-10 은 UNIX 체계가 경로조종기에 직접 접속된다고 가정한것이다.

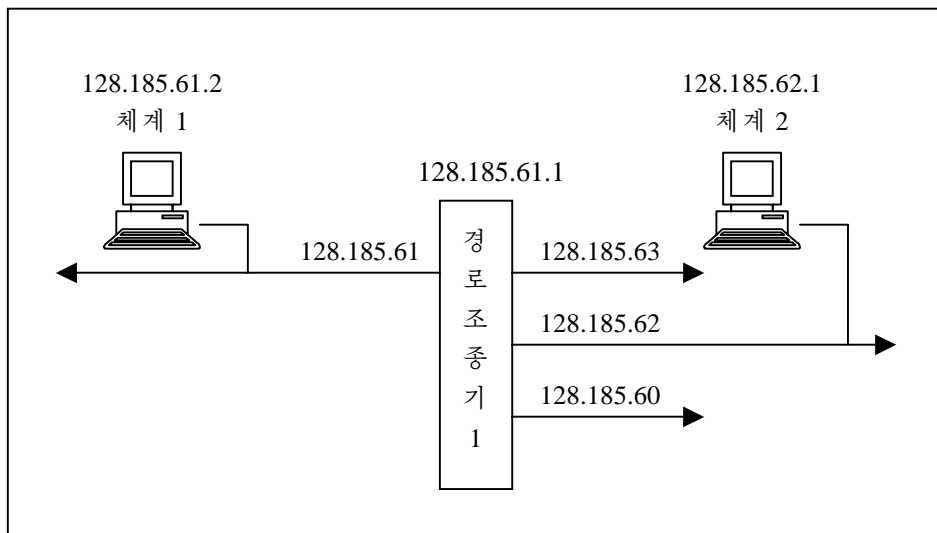



그림 16-10. UNIX 체계와 경로조종기의 실례

여기서 UNIX 체계는 토막 128.185.61 에 접속되여 있다. 이 토막의 주소는 가능한 부분망들을 가지는 클라스 "B"의 인터네트주소이다.

/etc/hosts 파일은 그 파일안에 매듭 ID2 를 가지는 UNIX 체계, 경로조종기, 이 토막 혹은 경로조종기의 다른 토막우에 있는 임의의 체계를 가질것을 요구한다.

만일 경로조종기가 단순히 모형화되여 있다면 토막 61 로부터 토막 60, 62, 63 우의 체계들까지 이음점이 없이 접속할수 있어야 한다. 경로조종기는 체계를 다른 토막들(60, 62, 63)에 있는 체계들에 접속할수 있게 모형화되여야 한다. 이런 단순한 망조작을 이음점없이 보장하려면 몇가지 특수한 모형화도 진행하여야 한다. 이 경우에 문제로 되는것은 60, 62, 63 우의 경로조종기의 다른 측면에 있는 체계들에 접속시키기 위하여 system1 을 읽을 때이다. 이런 문제를 해결하는데 보충적인 모형을 론의하기전에 먼저 /etc/hosts 파일을 보고 다음에 망의 상태를 보여 주는 효과적인 UNIX 의 일부 지령들을 리용하기로 한다. 아래에서는 UNIX 체계와 경로조종기만을 보여 주는 파일 /etc/hosts 를 주고 있다.

```
$ cat/etc/hosts
127.0.0.1    localhosts  loopback
128.185.61.1  router1        # router
128.185.61.2  system1        # UNIX  system  on  61
128.185.62.1  system2        # UNIX  system  on  62
```

이 호스트파일은 단순하며 system1 을 router1 과 system2 에 접속하도록 하고 있다. system1 로부터 system2 까지의 접속은 경로조종기를 통하여 수행된다.

ping

가장 공통적으로 리용되는 망지령들중의 한개의 지령인 ping 에 대하여 보기로 한다. 이 지령은 두개의 망성분들사이에 접속이 존재하는가 존재하지 않는가를 결정하기 위하여 리용된다. ping 은 초당 한번씩 호스트에 ICMP 통보파케트를 보내는 단순한 지령이다. 사용자는 망층아래 즉 세번째층에 있는 ICMP 를 다시 호출할수 있다. ping 은 Packet Internet Grouper 의 략자이다. ping 은 UNIX 변종들에서 약간씩 차이나는데 그 차이는 대체로 선택항목들이 하나도 없을 때 ping 이 생성하는 통보에서 나타난다.

ping 이 선택항목이 하나도 없이 쓰일 때 일부 체계들은 수행통보를 제공하며 다른 체계들은 체계가 살아 있다는것을 통보한다. 다음의 실례에서는 국부적체계와 austin 이라고 부르는 망우의 다른 체계사이의 접속검사에 대하여 보여 주고 있다.

```
martyp $ ping austin
austin is alive
martyp $
```

사용자는 대부분 UNIX 변종들에서 파케트의 크기와 반복회수를 조종할수 있다. 아

래에서는 HP-UX 에서 파케트의 크기를 4096 으로, 반복회수를 5 로 규정하는 실례를 보여 주고 있다.

```
# ping 12 4096 5
PING 12 : 4096 byte packets
4096 bytes from 10.1.1.12 : icmp_seq = 0 . time = 2 . ms
4096 bytes from 10.1.1.12 : icmp_seq = 1 . time = 2 . ms
4096 bytes from 10.1.1.12 : icmp_seq = 2 . time = 2 . ms
4096 bytes from 10.1.1.12 : icmp_seq = 3 . time = 2 . ms
4096 bytes from 10.1.1.12 : icmp_seq = 4 . time = 2 . ms
----12 PING Statistics----
5 packets transmitted , 5 packets received , 0% packet loss
round-trip (ms)  min/avg/max = 2 / 2 / 2
#
```

AIX 는 선택항목 -I 를 리용하여 파케트의 크기와 반복회수를 포함하여 구간을 규정하도록 한다. 다음 실례에서는 AIX 체계에서 이 선택항목들의 사용을 보여 주고 있다.

```
martyp $ ping -I 5 austin 4096 10
PING austin : 4096 data bytes
4104 bytes from austin (128.185.61.5) :  icmp_seq=0. time=8. ms
4104 bytes from austin (128.185.61.5) :  icmp_seq=l.  time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=2. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=3. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=4. time=8. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=5. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=6. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=7. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=8. time=9. ms
4104 bytes from austin (128.15.61.5) : icmp_seq=9. time=9. ms
---- austin PING Statistics----
10 packets transmitted, 10 packets received, 0% packet  loss
round-trip (ms) min/avg/max = 9 / 9 / 15
martyp $
```

우에서는 크기가 4096byte 인 파케트를 가지고 5s 에 한번씩 모두 10 번 ping austin 을 수행한 실례를 보여 주고 있다.

우의 실례를 거꾸로 설명하여 보자.

system1 과 경로조종기사이의 접속, system1 과 경로조종기의 다른 측면에 있는 다른

체계들사이의 접속을 하였다는것을 어떻게 알수 있겠는가? 지령 ping 을 사용하여 system1 이 router1 에 어떻게 접속되였는가를 다음과 같이 알수 있다.

```
$ ping router1
PING routerl: 64 byte packets
64 bytes from 128.185.61.2: icmp_seq=0. time=0. ms
64 bytes from 128.185.61.2: icmp_seq=1. time=0. ms
64 bytes from 128.185.61.2: icmp_seq=2. time=0. ms
```

우의 매 행들은 ping 이 수행된 장치로부터 귀환되는 반응들을 보여 주고 있다. 이것은 장치가 반응한다는것을 의미한다. 이 반응은 무한히 계속 얻을수 있으며 ping 을 끝내려면 ^c(Ctrl−c)를 입구해야 한다. 만일 아래에서 보여 준것처럼 결과가 하나도 출구되지 않았다면 그 어떤 반응도 없다는것이며 체계와 접속을 검사하는 장치사이에 일정한 문제가 발생하였다는것을 의미한다.

```
$ ping system2
PING router1: 64 byte packets
```

이러한 방법으로 우와 같은 통보를 얻을수 있다. ^c 로 ping 을 끝내면 보낸 파케트들과 받지 못한 파케트들을 보게 된다. 지령 ping 을 리용할 때 이러한 반응을 실지로 얻게 됨으로써 system1 과 router1 사이에서 문제가 있다는것을 알수 있다.

ping 이 실지로 망을 통하여 정보량을 송수신하기때문에 ping 은 망체계에서의 오유를 수정하기 위한 검사목적에만 리용되여야 한다. 련속적으로 수행되는 스크립트에서는 ping 을 리용하지 말아야 한다.

ping 을 리용하는 좋은 방안은 앞의 실례에서와 같이 기정값을 64byte 가 아니라 4096byte 의 크기로 파케트크기를 규정할 때와 ^c 를 입구하여 ping 을 끝내지 않고 종결전에 ping 전송회수를 계산할 때이다. 다음의 실례에서는 이에 대하여 보여 주고 있다.

```
$ ping routerl 4096  5
PING routerl: 64 byte packets
4096 bytes from 128.185.51.2: icmp_seq=0. time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=l.  time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=2. time=9. ms
4096 bytes from 128.185.51.2: icmp_seq=3. time=8. ms
4096 bytes from 128.185.51.2: icmp_seq=4. time=8. ms
```

통보를 전송하고 대답을 받는데 필요한 왕복시간은 64byte 만을 전송할 때의 시간보다 실지로 더 크다는것을 알아야 한다. 64byte 의 왕복전송시간은 망의 위상과 망교통과 같은 많은 인자들에 관계되지만 보통 0ms 에 가깝다는것을 알수 있다.

netstat

앞에서 부분망마스크에 대하여 고찰한바와 같이 한 호스트로부터 다른 호스트까지의 경로조종은 여러가지 방법으로 모형화될수 있다. 한 호스트로부터 다른 호스트로 정보를 얻는 길은 경로조종에 관계된다.

지령 netstat 를 통하여 경로조종에 관한 정보를 얻을수 있다. netstat 에서 선택항목 -r 는 보통 알고 싶어 하는 경로조종표를 보여 주며 선택항목 -n 은 이름이 아니라 수값으로 된 망주소를 제시하는데 리용된다. 선택항목 -v 로부터 부분망마스크와 같은 경로조종에 관계되는 추가적정보를 얻을수 있다. 다음 실례에서는 netstat 의 선택항목 -r(이것은 netstat 출구를 서술할 때 리용된다.), 선택항목 -rn, 선택항목 -rnv 의 결과를 대비적으로 고찰할수 있다.

```
# netstat -r
Routing tables
Dest/Netmask     Gateway      Flags  Refs    Use     Interface   Pmtu
o2               o2           UH      0  1890905     lo0         4136
o2               o2           UH      0      343     lan1        4136
o2               o2           UH      0        0     lan0        4136
10.1.1.0         o2           U       2        0     lan0        1500
10.1.1.0         o2           U       2        0     lan1        1500
127.0.0.0        o2           U       0        0     lo0         4136
default          10.1.1.1     UG      0        0     lan1        1500
#

# netstat -rn
Routing tables
Dest/Netmask     Gateway      Flags  Refs    Use    Interface    Pmtu
127.0.0.1        127.0.0.1    UH      0  1891016    lo0          4136
10.1.1.10        10.1.1.10    UH      0      343    lan1         4136
10.1.1.110       10.1.1.110   UH      0        0    lan0         4136
10.1.1.0         10.1.1.110   U       2        0    lan0         1500
10.1.1.0         10.1.1.10    U       2        0    lan1         1500
127.0.0.0        127.0.0.1    U       0        0    lo0          4136
default          10.1.1.1     UG      0        0    lan1         1500

# netstat -rnv
Routing tables
Dest/Netmask                  Gateway      Flags  Refs   Use     Interface  Pmtu
127.0.0.1/255.255.255.255     127.0.0.1     UH     0  1891036    lo0        4136
10.1.1.10/255.255.255.255     10.1.1.10     UH     0     343     lan1       4136
10.1.1.110/255.255.255.255  10.1.1.110      UH     0     0       lan0       4136
```

10.1.1.0/255.255.255.0	10.1.1.110	U	2	0	lan0	1500
10.1.1.0/255.255.255.0	10.1.1.10	U	2	0	lan1	1500
127.0.0.0/255.0.0.0	127.0.0.1	U	0	0	loo	4136
default/0.0.0.0	10.1.1.1	UG	0	0	lanl	1500

체계와 o2 를 보여 주는 첫번째와 두번째 출력는 3 개의 대면부를 가지고 있다. 즉 첫번째 대면부는 lo0 이라는 고리묶기대면부이다. 두번째 대면부는 .10 에 있으며 세번째 대면부는 .110 에 있다(-rn 출구로부터 볼수 있다.). 다음의 두개 행들은 망의 목적지인 10.1.1.0 을 보여 준다. 이 목적지는 .10 혹은 .110 기판을 통하여 접근될수 있다. 세번째 출구는 불필요한 정보를 제공한다. 마지막행은 기정경로에 대한 정보이다. 이 등록항목들은 더 직접적으로 경로가 찾아지지 않는다면 10.1.1.1 로 파케트를 보낸다는것을 의미한다.

netstat 에 의하여 경로조종기에 의한 정보가 제공된다. 선택항목 -r 는 경로조종방법에 대한 정보를 보여 주지만 이 지령의 다른 많은 선택항목들도 리용할수 있다. 이 출구에서 가장 기본적인것은 발생하는 경로조종의 형태를 정의하는 "Flags"이다. 아래에서는 UNIX 안내페지로부터 가장 일반적인 신호기들을 보여 주고 있다.

1=U	국부적인 호스트자체인 관문을 통한 망에로의 경로조종
3=UG	원격호스트인 관문을 통한 망에로의 경로조종
5=UH	국부적호스트자체인 관문을 통한 호스트에로의 경로조종
7=UGH	호스트인 원격관문을 통한 호스트에로의 경로조종

또한 경로조종정보와 대응되는 망통계자료를 얻으려면 두가지 형식의 netstat 를 리용할수 있다. 첫번째 형식은 netstat -i 로서 자동적으로 모형화된 대면부들의 상태를 보여준다. lan0 에 대한 개요를 얻으려면 이 지령을 리용하면 된다. netstat -i 는 작업상태의 망, 그 이름과 같은 lan0 의 좋은 개요를 준다.

다음의 실례는 Solaris 와 HP-UX 체계에서 netstat -i 의 결과를 보여 준다.

netstat -i

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs	Coll
ni0*	0	none	none	0	0	0	0	0
nil*	0	none	none	0	0	0	0	0
1o0	4608	loopback	127.0.0.1	232	0	232	0	0
lan0	1500	169.200.112	169.200.112.2	3589746	2	45630	0	104

netstat -i

Name	Mtu	Network	Address	Ipkts	Opkts
lani	1500	10.1.1.0	o2	59935480	163641547
lan0	1500	10.1.1.0	o2	139173	12839358
1o0	4136	127.0.0.0	o2	1892333	1892345

#

아래에서는 netstat 실례의 마당들을 해설하고 있다.

Name	망대면부의 이름. 실례에서는 lan0이다.
MTU	대면부기판으로 보내는 최대파케트크기인 "최대전송단위".
Network	대면부기판이 접속되는 LAN의 망주소(169.200).
Address	체계의 호스트이름. 이것은 /etc/hosts 파일에서 나타나는것과 같은 체계에 대한 기호적이름이다.

통계정보는 다음과 같은것을 포함한다.

Ipkts	대면부기판에 의하여 받는 파케트개수. 우의 실례에서는 Lan0이다.
Ierrs	대면부기판에 의하여 들어온 파케트들에서 발견된 오유개수.(일부 UNIX방안들에서 가능하다.)
Opkts	대면부기판에 의하여 전송되는 파케트의 개수.
Oerrs	대면부기판에 의하여 파케트들이 전송되는 동안에 발견되는 오유개수.(일부 UNIX방안들에서 가능하다.)
Collis	파케트교통에서 발생하는 충돌의 개수. (일부 UNIX 방안들에서 가능하다.)

netstat는 매듭이 마지막시동이 있은 때로부터 축적된 자료를 제공한다. 자료의 저축에는 많은 시간이 소비된다. 만일 유용한 통계적정보를 보는데 관심이 있다면 다른 선택항목들도 netstat에서 리용할수 있다. 사용자는 통계적자료를 받을수 있는 구간을 규정할수도 있다. 첫번째 등록항목이 체계가 마지막으로 시동한 때로부터의 모든 자료를 보여주기때문에 그것은 보통 무시된다. 그러므로 자료는 체계가 작업하지 않고 있을 때의 부차시간도 포함한다. 보통 사용자들은 체계가 작업하고 있을 때의 자료를 보게 된다. 다음의 netstat실례에서는 Solaris체계에서 5s에 한번씩 망대면부정보를 제공해 주고 있다.

\# **netstat -I lan0 5**

(lano)-> input		output			(Total)-> input		output		
packets	errs	packets	errs	colls	packets	errs	packets	errs	colls
3590505	2	45714	0	104	3590737	2	45946	0	104
134	0	5	0	0	134	0	5	0	0
174	0	0	0	0	174	0	0	0	0
210	0	13	0	0	210	0	13	0	0
165	0	0	0	0	165	0	0	0	0
169	0	0	0	0	169	0	0	0	0
193	0	0	0	0	193	0	0	0	0
261	0	7	0	0	261	0	7	0	0
142	0	8	0	0	142	0	8	0	0
118	0	0	0	0	118	0	0	0	0
143	0	0	0	0	143	0	0	0	0
149	0	0	0	0	149	0	0	0	0

이 실례에서는 LAN 대면부에서 일어 나는 여러개의 결과를 보여 준다. 이미 언급한 것처럼 첫번째 출구가 긴 시간주기의 정보를 포함하기때문에 그것을 무시할수 있다. 이것은 망이 작업하지 않고 있을 때의 시간을 포함하고 있으므로 그 자료는 중요하지 않다.

다음 netstat 실례에서는 HP-UX lli 체계에서 5s 에 한번씩 망대면부정보를 제공해 주고 있다.

```
# netstat  -I  1an0  5
(lan0)->  input     output      (Total)->  input     output
          packets   packets                packets   packets

          139185    12841621               61968131  178375605
          139185    12841714               61968172  178375698
          139185    12841810               61968213  178375794
          139185    12841877               61968247  178375861
          139185    12841912               61968265  178375896
          139185    12842095               61968358  178376079
          139187    12842244               61968413  178376240
          139189    12842352               61968470  178376360
          139189    12842453               61968525  178376461
          139190    12842482               61968565  178376498
          139190    12842539               61968594  178376555
          139190    12842671               61968667  178376699
```

사용자는 -I interface 를 리용하여 통계자료가 출구되는 망대면부를 규정할수 있다. 실례에서는 -I lan0 이다. 실례에서는 5s 에 한번씩 리용되고 있다.

netstat 의 다른 하나의 리용은 망소케트의 상태를 보여 주는것이다.

netstat -a 는 규약, 대기렬, 국부적 및 원격적주소, 규약상태들의 목록을 생성시킨다. 이 모든 정보들은 다음 실례에서 보여 준것처럼 현재의 통신들을 보는데 편리하다.

```
# netstat -a
Active Internet connections (including servers)
Proto  Recv-Q  Send-Q  Local Address     Foreign Address        (state)
tcp    0       2       systeml.telnet    atlm0081.atl.hp..1319  ESTABLISHED
tcp    0       0       *.1095            *.*                    LISTEN
tcp    0       0       *.psmond          *.*                    LISTEN
tcp    0       0       *.mcsemon         *.*                    LISTEN
tcp    0       0       localhost.8886    localhost.1062         ESTABLISHED
tcp    0       0       localhost.1062    localhost.8886         ESTABLISHED
tcp    0       0       *.8886            *.*                    LISTEN
tcp    0       0       *.8887            *.*                    LISTEN
```

tcp	0	0	*.1006	*.*	LISTEN
tcp	0	0	*.978	*.*	LISTEN
tcp	0	0	*.22370	*.*	LISTEN
tcp	0	0	*.389	*.*	LISTEN
tcp	0	0	*.8181	*.*	LISTEN
tcp	0	0	*.1054	*.*	LISTEN
tcp	0	0	*.1053	*.*	LISTEN
tcp	0	0	*.diagmond	*.*	LISTEN
tcp	0	0	*.1045	*.*	LISTEN
tcp	0	0	*.1038	*.*	LISTEN
tcp	0	0	*.135	*.*	LISTEN
tcp	0	0	*.smtp	*.*	LISTEN
tcp	0	0	*.1036	*.*	LISTEN
tcp	0	0	*.appconn	*.*	LISTEN
tcp	0	0	*.spc	*.*	LISTEN
tcp	0	0	*.dtspc	*.*	LISTEN
tcp	0	0	*.recserv	*.*	LISTEN
tcp	0	0	*.klogin	*.*	LISTEN
tcp	0	0	*.kshell	*.*	LISTEN
tcp	0	0	*.chargen	*.*	LISTEN
tcp	0	0	*.discard	*.*	LISTEN
tcp	0	0	*.echo	*.*	LISTEN
tcp	0	0	*.time	*.*	LISTEN
tcp	0	0	*.daytime	*.*	LISTEN
tcp	0	0	*.printer	*.*	LISTEN
tcp	0	0	*.auth	*.*	LISTEN
tcp	0	0	*.exec	*.*	LISTEN
tcp	0	0	*.shell	*.*	LISTEN
tcp	0	0	*.login	*.*	LISTEN
tcp	0	0	*.telnet	*.*	LISTEN
tcp	0	0	*.ftp	*.*	LISTEN
tcp	0	0	*.795	*.*	LISTEN
tcp	0	0	*.792	*.*	LISTEN
tcp	0	0	*.*	*.*	CLOSED
tcp	0	0	*.787	*.*	LISTEN
tcp	0	0	*.783	*.*	LISTEN
tcp	0	0	*.779	*.*	LISTEN
tcp	0	0	*.portmap	*.*	LISTEN
tcp	0	0	*.2121	*.*	LISTEN
udp	0	0	*.1127	*.*	

udp	0	0	*.177	*.*
udp	0	0	*.1003	*.*
udp	0	0	*.*	*.*
udp	0	0	*.*	*.*
udp	0	0	*.*	*.*
udp	0	0	*.*	*.*
udp	0	0	*.nfsd	*.*
udp	0	0	*.976	*.*
udp	0	0	*.22370	*.*
udp	0	0	*.1097	*.*
udp	0	0	*.1095	*.*
udp	0	0	*.1079	*.*
udp	0	0	*.135	*.*
udp	0	0	*.*	*.*
udp	0	0	*.1045	*.*
udp	0	0	*.snmp	*.*
udp	0	0	*.1040	*.*
udp	0	0	*.tftp	*.*
udp	0	0	*.chargen	*.*
udp	0	0	*.discard	*.*
udp	0	0	*.echo	*.*
udp	0	0	*.time	*.*
udp	0	0	*.daytime	*.*
udp	0	0	*.ntalk	*.*
udp	0	0	*.bootps	*.*
udp	0	0	*.1023	*.*
udp	0	0	*.787	*.*
udp	0	0	*.798	*.*
udp	0	0	*.797	*.*
udp	0	0	*.1037	*.*
udp	0	0	*.*	*.*
udp	0	0	*.1036	*.*
udp	0	0	*.1035	*.*
udp	0	0	*.777	*.*
udp	0	0	*.portmap	*.*
udp	0	0	*.1034	*.*
udp	0	0	*.syslog	*.*
udp	0	0	*.2121	*.*

Active UNIX domain sockets

Address	Type	Recv-Q	Send-Q	Inode	Conn	Refs	Nextref	Addr

```
bb9c00   stream  0  0  af9000        0  0       0       /tmp/.AgentSoA
ced700   dgram   0  0  c99400        0  0       0       /opt/dcelocalr
ce9e00   dgram   0  0  d23000        0  0       0       /opt/dcelocalr
b0d200   dgram   0  0  b87000        0  0       0       /opt/dcelocalr
997a00   stream  0  0  b84800        0  0       0       /opt/dcelocal1
b24e00   dgram   0  0  b84000        0  0       0       /opt/dcelocal1
d59400   dgram   0  0  b66400        0  0       0       /var/tmp/psb_t
d85c00   dgram   0  0  b67000        0  0       0       /var/tmp/psb_t
c8b200   dgram   0  0  b12000        0  0       0       /opt/dcelocalr
c8b400   stream  0  0  b78400        0  0       0       /opt/dcelocal5
c8b300   dgram   0  0  b78000        0  0       0       /opt/dcelocal5
c90900   dgram.  0  0  d22400        0  0       0       /opt/dcelocalr
c78c00   dgram   0  0  bal000  c4a180  0       0       /opt/dcelocal0
ble900   dgram   0  0  9a4400        0  c32e80  0       /opt/dcelocald
d64100   stream  0  0  d24c00        0  0       0       /opt/dcelocal5
9e1600   dgram.  0  0  9a4000  d4d940  0       0       /opt/dcelocal2
d64200   dgram   0  0  cfc800        0  c32c80  0       /opt/dcelocal9
d12d00   dgram   0  0  cfc000  c32c00  0       0       /opt/dcelocal1
c5ee00   stream  0  0  blc000        0  0       0       /opt/dcelocal4
d19d00   dgram   0  0  ce4800        0  0       0       /opt/dcelocald
cfoc00   dgram   0  0  a92800        0  afl5c0  0       /opt/dcelocal7
d2d600   dgram   0  0  a93800  c32c00  0       d4db80  /opt/dcelocal0
c9b900   dgram   0  0  a93c00        0  0       0       /opt/dcelocald
d6c800   stream  0  0  ba3000        0  0       0       /var/opt/OV/sT
#
```

우의 실례에서는 많은 정보들을 제시하고 있다. 만일 마당들에 대한 상세한 설명을 요구한다면 이 장의 마감에 있는 지령소개를 참고하면 된다.

첫 행은 상태 ESTABLISHED 들을 가지는 Local Address system1.telnet 에 대한 proto tcp 를 보여 준다. 이것은 이 체계가 시동될 때 접속된것이다. 현재 사용자는 netstat 를 수행시키는 열린 전화망가입기간을 가진 system1 에서 작업하고 있다.

나머지 tcp 규약들의 구체례들이 렬거되여 있다. 이것은 LISTEN 에 의하여 지적된것과 같이 들어 오는 접속들에 대한 목록들이다. Foregin Addres 마당에서 통용기호를 리용하고 있는데 그 통용기호는 접속이 진행될 때 주소를 포함하게 된다. ESTABLISHED 에서 지적된바와 같이 만들어 진 접속들중의 하나를 취하고 있다.

Recv-Q 와 Send-Q 로서 보여 준 모든 보내고 받는 대기렬은 0 으로 비여 있다는것을 지적하고 있다.

결과의 끝에 있는 UNIX 령역소케트들은 NFS 와 같은 여러가지 봉사에 필요한 자료흐름형식의 접속들이다.

이 결과는 UNIX 체계에서의 망관리에서 제기되는 작업에 대한 총적인 개괄을 주고 있다. 망작업과 접속은 UNIX 체계의 시초로부터 그것의 가장 발전된 측면들까지 포함하여 진행된다.

route

netstat 에 의하여 표시되는 정보는 체계에 대한 경로조종표이다. 일부는 체계가 기동되거나 망대면부가 시동될 때 지령 ifconfig 에 의하여 자동적으로 창조된다. 체계에 직접 접속되지 않은 망과 호스트들에 대한 경로조종은 지령 route 에 의하여 진행된다.

경로조종의 변경은 U 로부터 UG 까지 신호기들을 변경시키는것처럼 쉽게 할수 있다.

$ /usr/sbin/route add default 128.185.61.1 3

첫번째로는 지령 route 을 준다. 두번째로는 추가하려는 경로 혹은 제거하려는 경로를 규정하여야 한다. 세번째로는 목적지를 규정하여야 한다. 목적지로는 호스트이름, 망이름, IP 주소 혹은 실례에서 보여 준바와 같이 관문을 규정하는 기정값이 될수 있다. 네번째로는 목적지를 탐색하기 위한 관문을 주어야 한다. 우의 실례에서는 IP 주소가 리용되였지만 그것은 호스트의 이름으로 될수도 있다. 실례의 제일 마지막 3 은 관문이 국부적호스트인가 혹은 원격적관문인가를 규정하는데 리용되는 계수기에 대응한다. 만일 관문이 국부적호스트이면 계수기의 값은 0 이다. 만일 관문이 원격호스트이면 계수기는 0 보다 큰 값을 리용한다. 이것은 Flags 에서는 UG 에 대응한다. Flags 에서 기정의 경로를 추가함으로써 경로조종표를 변경시키고 있다. 이 지령을 리용하면 경로조종기의 다른 측면에 있는 체계들과 system1 이 대화하도록 시도하면서 제기하였던 문제를 해결할수 있다(그림 16-10 을 상기하시오.).

선택항목 add 로서 /usr/sbin/route 를 리용하기전에 현재 작업하지 않으면서 존재하고 있는 기정경로를 제거하기 위하여 선택항목 delete 를 리용할수 있다.

지령 route 는 보통 체계의 시동파일들중의 한개에서 나타날수 있으며 그것은 체계가 기동할 때마다 수행된다. 그리하여 체계가 시동할 때마다 정확한 접속정보가 놓이도록 담보된다.

ifconfig

지령 ifconfig 는 LAN 대면부에 대한 추가적정보를 제공한다. 다음의 실례에서는 망대면부의 모형을 보여 주고 있다.

$ **/etc/ifconfig lan0**

lan0: flags=863<UP，BROADCAST，NOTRAILERS，RLUNNING>

inet 128.185.61.2 netmask ffff0000 broadcast 128.185.61.255

이 실례로부터 대면부가 주소 128.185.61.2 를 가지며 망마스크 ffff0000 을 가진다는것을 볼수 있다. 망대면부는 le0 와 같은 다른 이름을 가질수도 있다.

망대면부에 주소를 부과하거나 망대면부의 파라메터들을 설정할 때 망대면부의 상태를 알기 위하여 ifconfig 를 리용할수 있다. 망주소는 이미 언급된것처럼 A, B, C 클라스안에 놓인다. LAN 대면부를 형성하기전에 망의 클라스를 확인해 볼수 있다. 이 실례에서는 클라스 B 에 속하는 망을 보여 주고 있다. 클라스 B 의 망마스크는 전형적인 ffff0000 이며 클라스 C 의 망마스크는 ffff00 이다. 망마스크는 망을 더 작은 망으로 분할할 때 얼마나 많은 주소를 예약할수 있는가를 결정하는데 리용된다. 망마스크는 우에서 보여 준것처럼 16 진으로 표시될수 있지만 파일 /etc/hosts 에서와 같이 10 진으로 표시될수도 있다. 아래에서는 대면부를 모형화하기 위한 지령 ifconfig 을 보여 준다.

```
$/etc/ifconfig  lan0  lan0  inet  128.185.61.2  netmask  255.255.0.0
```

- 255.255.0.0 은 이미 보여 준 클라스 B 의 부분망마스크 fff0000 에 대응한다.
- lan0 은 모형화될 대면부이다.
- inet 는 이 체계에 제공되는 주소족이다.
- 128.185.61.2 는 system1 에서 LAN 대면부의 주소이다.
- netmask 는 망을 부분망으로 어떻게 구분하는가를 보여 준다.
- 255.255.0.0 은 클라스 B 의 마스크로서 ffff0000 와 동일하다.

망의 상태를 효과적으로 보려면 netstat, ping, ifconfig 를 리용하면 된다. ifconfig, route 와 /etc/hosts 는 망을 설치하는데 리용되며 다른 변경과 식별하여야 한다. 부분망은 현재와 미래까지 다 고려하여 망을 설치할수 있는 유연성을 가지고 있다. 단순한 망에서는 지령들이 다 요구되지 않으며 복잡한 부분망에서는 모든 지령들이 다 요구된다. 복잡한 망이나 망설치에서 난관에 부닥칠 때마다 이러한 지령들을 서로 결합하여 리용하여야 한다. 또한 망계획화는 UNIX 체계설치에서 중요한 부분으로 된다.

이 장의 전반에서 리용되는 지령들은 매 체계관리자의 도구창에서 볼수 있다. 망은 UNIX 체계의 리용에서 생명이며 따라서 이 부문에 대한 지식을 가져야 체계전반을 쉽게 리해할수 있으며 보다 효과적으로 리용할수 있다.

rpcinfo

사용자들은 다른 체계에 대한 등록부를 NFS 로 설정할수 있으며 자기의 체계에 없는 기능들을 수행할수 있다. 데몬을 수행시켜 어떤 기능들이 가능한가를 평가할수 있다. rpcinfo 는 국부적체계까지 포함하여 체계에 대한 원격적틀호출(RPC)을 가능하게 한다. 이때 지령은 rpc -p system_name 의 형식을 가진다.

다음의 실례는 국부적체계에서의 rpcinfo -p 의 리용을 보여 준다.

```
# rpcinfo   -p
   program      vers       proto       port        service
   100000       2          tcp         111         portmapper
```

100000	2	udp	111	portmapper
100024	1	udp	777	status
100024	1	tcp	779	status
100021	1	tcp	783	nlockmgr
100021	1	udp	1035	nlockmgr
100021	3	tcp	787	nlockmgr
100021	3	udp	1036	nlockmgr
100020	1	udp	1037	llockmgr
100020	1	tcp	792	llockmgr
100021	2	tcp	795	nlockmgr
100068	2	udp	1040	cmsd
100068	3	udp	1040	cmsd
100068	4	udp	1040	cmsd
100068	5	udp	1040	cmsd
100083	1	tcp	1036	ttdbserver
100005	1	udp	976	mountd
100005	1	tcp	978	mountd
100003	2	udp	2049	nfs
150001	1	udp	1003	pcnfsd
150001	2	udp	1003	pcnfsd
150001	1	tcp	1006	pcnfsd
150001	2	tcp	1006	pcnfsd

체계에서 중요한 기능들에 대하여 많은 데몬들이 수행된다. mountd 는 봉사기가 NFS 파일체계를 설정하도록 지적한다. 이 기능에 대한 데몬도 수행된다. 추가적으로 pchfsd 는 Windows 에 기초한 NFS 의 접근을 보장한다.

arp

IP 주소와 그것에 대응하는 MAC 주소의 목록을 관리하는 도구는 arp 고속완충기에 있다. 주소는 일시적으로 고속완충기에 보존되므로 어떤 주소가 방금 취해졌는가를 보려면 지령 arp 를 리용하여야 한다. 아래에서는 지령 arp 의 리용실례를 보여 주고 있다.

arp -a
o2（10.1.1.10）at 0:10:83:f7:a2:fB ether
11（10.1.1.11）at 0:10:83:f7:2e:d0 ether
63.88.85.1（63.88.85.1）at 0:30:94:b0:bB:a0 ether
13（10.1.1.200）at 0:10:83:fc:92:88 ether
tapel（10.1.1.14）at 0:10:83:f7:e:32 ether

tapel（10.1.1.14）at 0:10:83:f7:e:32 ether
tapel（10.1.1.14）at 0:10:83:f7:e:32 ether
tapel（10.1.1.14）at 0:10:83:f7:e:32 ether
63.88.85.18（63.88.85.18）-- no entry

현재의 지령 arp 는 지령 -a 과 함께 표시된다. 선택항목 -s 도 리용할수 있다.

lanadmin

lanadmin 은 망기판을 보고 그것에 대한 조직을 진행하는데 리용된다. 아무런 선택항목도 없이 lanadmin 을 리용하면 다음 실례에서 보여 준바와 같이 대화방식의 대면부로 작업할수 있다.

lanadmin

LOCAL AREA NETWORK ONLINE ADMINISTRATION, Version 1.0

Copyright 1994 Hewlett Packard Company.
All rights are reserved.

Test Selection mode.
lan = LAN Interface Administration
menu = Display this menu
quit = Terminate the Administration
terse = Do not display command menu
verbose = Display command menu
Enter command: **lan**

LAN Interface test mode. LAN Interface PPA Number = 0

clear = Clear statistics registers
display = Display LAN Interface status and statistics registers
end = End LAN Interface Administration, return to Test Selection
menu = Display this menu
ppa = PPA Number of the LAN Interface
quit = Terminate the Administration, return to shell
reset = Reset LAN Interface to execute its selftest
specific = Go to Driver specific menu

```
Enter command: d

                    LAN INTERFACE STATUS DISPLAY

PPA Number                          = 0
Description              = lan0 Hewlett-Packard 10/100 TX Half-Duplex TT = 1500
Type (value)                        = ethernet-csmacd(6)
MTU Size                            = 1500
Speed                               = 100000000
Station Address                     =         0x1083ffcaae
Administration Status (value)       = up(l)
Operation Status (value)            = down(2)
Last Change                         = 237321866
Inbound Octets                      = 0
Inbound Unicast Packets             = 0
Inbound Non-Unicast Packets         = 0
Inbound Discards                    = 0
Inbound Errors                      = 0
Inbound Unknown Protocols           = 0
Outbound Octets                     = 820
Outbound Unicast Packets            = 20
Outbound Non-Unicast Packets        = 0
Outbound Discards                   = 1
Outbound Errors                     = 0
Outbound Queue Length               = 0
Specific                            = 655367

Press <Return> to continue
```

우의 실례에서는 lanadmin 을 실행하여 lan 대면부를 조직하며 그 대면부에 대한 정보를 표시하고 있다.

lanadmin 은 또한 선택항목 -M 과 -s 로서 MTU 혹은 lan 대면부의 속도를 변경시키는 것과 같은 과제를 수행할수 있다.

ndd

ndd 는 망을 조화롭게 수행시키고 망파라메터들에 대한 정보를 보는데 리용된다. ndd 로서 제공되는 모든 조화파라메터들에 대한 정보를 보려면 ndd -h supproted 라고 입

력해야 한다. 선택항목 -get 를 리용하여 파라메터의 값을 읽을수 있으며 선택항목 -set 로서 파라메터의 값을 설정할수 있다.

nslookup

nslookup 은 호스트이름을 IP 주소로 만들기 위하여 리용된다. nslookup hostname 과 nslookup 을 입력하면 호스트이름을 해결하기 위하여 /etc/resolv.conf 파일 혹은 /etc/hosts 에 접근할수 있다. 다음의 실례에서는 /etc/hosts 를 리용하여 system1 의 IP 주소를 생성하는것을 보여 주고 있다.

```
#  nslookup  12
Using  /etc/hosts  on:  13

looking  up  FILES
Name:      12
Address:   10.1.1.12
#
```

아무런 지령행인수가 없이 이 지령을 리용하여 nflookup 을 대화방식으로 수행시킬수 있다. 다음의 실례에서는 이러한 대화방식을 수행시키는 과정과 지령에 대한 정보를 얻기 위하여 help 를 리용하는것을 보여 주고 있다.

```
#  nolookup   12
> help
NAME               - print address information about NAME

IP-ADDRESS         - print hostname information about IP-ADDRESS
policy             - print switch policy information
server NAME        - set default server to NAME, using current default server
lserver NAME       - set default server to NAME, using initial server
set OPTION         - sets the OPTION
    all            - print options, current server and host
    (no) swtrace   - print lookup result and lookup switch messages
>
```

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 약간 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

ftp

ftp – 파일전송프로그람을 위한 대면부.

---

ftp(1)

이름

ftp – 파일전송프로그람

형식

ftp [-g] [-i] [-n] [-r] [B size] [server host]

해설

ftp는 파일전송규약에 대한 사용자대면부이다. ftp는 국부적의뢰기호스트와 원격봉사기호스트사이의 망접속에 대한 파일들을 복사한다. ftp는 의뢰기호스트에서 수행된다.

선택항목들

-g 파일이름의 제거를 불가능하게 한다(아래에 있는 glob 지령을 참고). 이 선택항목이 규정되지 않으면 제거가 가능하다.

-i 여러개의 파일지령들에 의한 대화방식의 작업을 불가능하게 한다(아래에 있는 prompt 지령을 참고). 이 선택항목이 규정되지 않으면 대화방식이 가능하다.

-n 자동가입을 불가능하게 한다(아래에 있는 open 지령을 참고). 이 선택항목이 규정되지 않으면 자동가입이 가능하다.

-v 불필요한 출구도 가능하게 한다(아래에 있는 verbose 지령을 참고). 이 선택항목이 규정되지 않으면 ftp는 말단이 표준입구장치인 때에만 불필요한 출구를 현시한다.

-B 1024Byte의 블로크크기로 자료소케트의 완충기크기를 설정한다. 이 값은 1-64까지의 옹근수가 될수 있다.(기정값은 56이다.)

주의 : 완충기크기가 크면 빠른 련결(FDDI) 에 대한 ftp 의 성능을 개선할수 있지만 느린 련결(X.25)에 대한 접속시간은 길어 진다.

ftp 가 통신하는 봉사기호스트의 이름은 지령행에서 규정될수 있다. 봉사기호스트가 규정되면 즉시에 ftp 는 직접 봉사기호스트에 접속을 연다(아래에 있는 지령 open 을 참고). 그 밖의 경우에 ftp 는 사용자로부터의 지령들을 대기한다.

파일전송규약은 종류, 방식, 형식, 구조를 위한 파일전송파라메터들을 규정한다. ftp 는 ASCII, 2 진, tenex 파일전송규약종류들을 제공한다. ASCII 는 기정의 FTP 종류이다.(그것은 ftp 가 2 개의 류사한 체계들사이의 접속을 설정할 때마다 보다 효과적인 2 진형으로 자동적으로 절환한다.) ftp 는 기정으로 흐름식파일전송파라메터방식, 인쇄되지 않는 형식, 파일의 구조를 보장한다.

지령들

ftp 는 다음과 같은 지령들을 봉사한다. 공백이 포함된 지령항목들은 ""안에 포함되여야 한다(실례로 "argment with embedded spaces").

! [command [args]]
: 국부적호스트에서 쉘을 기동시킨다. SHELL 환경변수는 기동되는 쉘프로그람을 규정한다. ftp 는 SHELL 이 정의되지 않았다면 /usr/bin/sh 를 기동한다. command 가 규정되면 쉘은 지령 command 를 수행하고 ftp 에 귀환한다. 그밖의 경우에는 대화식쉘이 기동된다. 쉘이 끝날 때 ftp 에 귀환한다.

$ macro-name [args]
: 지령 macdef 으로 정의된 마크로인 macro-name 을 수행한다. 인수들은 마크로에 전달된다.

account [passwd]
: 가입이 성과적으로 완성되면 원격체계에서 요구되는 자원접근을 위한 보충적통과암호를 제공한다. 만일 인수가 없으면 사용자는 비표시입력방식으로 가입통과암호의 입력재촉문을 재촉받는다.

append local-file [remote-file]
: remote-file 의 끝에 local-file 을 복사한다. 만일 remote-file 이 규정되여 있지 않으면 local-file 은 ntrans 혹은 nmap 설정으로 변경된후에 원격파일의 이름으로 리용된다.

ascii　파일전송형을 망 ASCII 로 설정한다. 이것은 기정형으로 된다.

bell　매 파일전송을 완성한 후에 경고를 내보낸다.

binary　파일전송형을 2 진으로 설정한다.

bye　만일 접속이 열렸다면 봉사기호스트접속을 닫고 끝낸다. EOF 기호

가 입구되면 종결하고 가입기간에서 탈퇴한다.

case　지령 mget 을 수행할동안 원격콤퓨터의 파일이름의 대소문자변환을 고정한다. case 가 on 일 때 (기정은 off 이다.)원격콤퓨터파일이름들에서 모든 대문자들은 소문자로 변경되여 국부적등록부에 씌여 진다.

cd remote-directory

봉사기호스트에서 작업등록부를 원격등록부로 설정한다.

cdup　봉사기호스트의 작업등록부를 현재 원격작업등록부의 어미등록부로 설정한다.

chmod mode file-name

원격체계에서 file-name 파일들의 허락방식을 mode 로 변경한다.

close　봉사기호스트와의 접속을 종결한다. 지령 close 은 ftp 를 종결하지 않는다. 임의의 정의된 마크로들은 지워진다.

cr　ascii 형파일이 탐색될동안 제거할 CR 를 고정한다. 기록들은 ASCII 형파일을 전송할 때에 CR/LF 로 지적된다. cr 가 on 일 때 (기정이다.)UNIX 는 CR 를 한개의 문자로 된 LF 기록분리기호로 고찰하면서 제거한다. UNIX 를 리용하지 않는 원격체계에서 기록들은 한개의 문자로 된 LF 를 포함할수 있다. ascii 형전송을 할 때 이 LF 는 CR 가 off 로 될 때에만 기록분리기호로 식별될수 있다.

delete remote-file

remote-file 을 제거한다. remote-file 은 빈 등록부도 될수 있다.

dir [remote-directory] [local-file]

표준출력장치 혹은 local-file 에 remote-directory 를 쓴다. 만일 local-file 도, remote-directory 도 규정되지 않으면 표준출력장치에 remote-directory 의 목록을 출구한다. 만일 대화적방식으로 작업하면 ftp 는 마지막인수가 실지로 dir 출력의 목적파일인가를 검증하기 위한 통보를 사용자에게 내보낸다. 생략된 기호들은 항상 확장된다.

disconnect

close 와 동일하다.

form format

파일전송형식을 format 로 설정한다. 제공되는 형식은 인쇄되지 않는것뿐이다.

get remote-file [local-file]

remote-file 을 local-file 에 복사한다. 만일 local-file 이 규정되지 않으면 ftp 는 국부파일이름으로써 규정된 원격파일이름을 리용한다. 현재의 case, ntrans, namp 설정으로 변경될수 있다.

glob　파일이름생략을 고정한다. 파일이름을 제거할수 있을 때 ftp는 파일과 등록부이름들에서 csh(1)의 메타기호들을 확장한다. 이러한 기호들로서는 *, ?, [,], ~, {, }이 있다. 봉사기호스트는 원격파일과 등록부이름을 전개한다. 제거된 메타기호들은 항상 지령 ls와 dir로 확장되여야 한다. 만일 제거할수 있으면 메타기호들은 mdelete, mdir, mget, mls, mput와 같은 파일지령들에서도 확장된다.

hash　전송되는 매 1024byte당 기호 ＃이 인쇄되도록 한다.

help [command]

지령 ftp를 호출한 지령 ftp에 대한 통보를 인쇄한다. 만일 지령 ftp가 규정되지 않으면 모든 ftp지령의 목록을 인쇄한다.

idle [seconds]

원격봉사기에 멎어 있는 시계를 seconds초로 설정한다. 만일 seconds가 생략되면 ftp는 현재의 멎어 있는 시계를 출력한다.

lcd [load-directory]

local-directory로 국부작업등록부를 설정한다. 만일 local-directory가 규정되지 않으면 사용자의 국부홈등록부로 국부작업등록부를 설정한다.

ls [remote-directory] [local-file]

remote-directory의 목록을 local-file에 쓴다. 목록은 봉사기가 포함시키려고 선택한 임의의 체계종속성정보를 포함한다. 실례로 대부분 UNIX 체계는 지령 ls -l로부터 출구를 생성한다(nlist를 참고). 만일 remote-directory 혹은 local-file이 규정되지 않으면 원격작업등록부를 목록화한다. 만일 제거할수 있으면 제거한 메타기호들은 확장된다.

macdef macro-name

마크로를 정의한다. 순차적행들은 마크로 macro-name으로 기억된다. 빈행은 마크로입구방식을 종결한다. 16개까지의 마크로로 제한되며 정의된 마크로에서 총 4096개까지의 기호들로 제한된다. 마크로들은 지령 close이 수행될 때까지 정의된대로 남아 있게 된다. 마크로처리자는 전문적기호로서 $와 \를 해석한다. $뒤에 놓이는 수들은 마크로발생지령행에서 대응하는 인수로 교체된다. $뒤에 있는 i는 모든 수행되는 마크로가 순환되여야 한다는것을 마크로처리자에게 신호한다. 첫번째 통과에서 $1은 마크로발생지령행에서 첫 인수로 교체되고 두번째 통과에서 그것은 두번째 인수로 교체된다.
\뒤에서 임의의 기호는 그 기호로 교체된다. $의 특수한 처리를 막기 위하여 \이 리용된다.

mdelete [remote-files]

remote-files을 제거한다. 만일 제거할수 있으면 제거된 메타기호들

은 확장된다.

mdir remote-file local-file

remote-file 의 부분을 local-file 에 쓴다. 만일 제거할수 있으면 제거된 메타기호들은 확장된다. 만일 대화방식에서 작업하면 ftp 는 마지막인수가 mdir 출력을 위한 실지의 목적파일이라는것을 확정하기 위하여 사용자에게 통보를 내보낸다.

mget remote-files

remote-files 을 국부체계에 복사한다. 만일 제거할수 있으면 제거된 메타기호들은 확장된다. 결과의 국부파일이름은 case, ntrans, nmap 설정에 따라 처리된다.

mkdir directory-name

원격등록부이름을 조성한다.

mls remote-file local-file

local-file 에 remote-file 의 생략된 목록을 쓴다. 만일 제거할수 있으면 제거된 메타기호들은 확장된다. 만일 대화방식에서 작업하면 ftp 는 마지막인수가 mls 출력을 위한 실지의 목적파일이라는것을 확정하기 위하여 사용자에게 통보를 내보낸다.

mode [mode-name]

FTP 파일전송방식을 mode-name 으로 설정한다. 제공된 방식은 흐름방식뿐이다.

modtime remote-file

remote-file 의 마지막변경시간을 보여 준다.

mput local-files

원격체계에 국부적체계로부터 local-files 들을 복사한다. 원격파일들은 ntrans 와 nmap 설정에 따라 처리된 국부파일과 같은 이름을 가진다. 만일 제거할수 있으면 제거된 기호만 확장된다.

newer file-name

원격파일의 변경시간이 현재체계에서 그 파일의 최근 시간보다 더 이전일 때에만 파일을 읽는다. 만일 파일이 현재 체계에 존재하지 않으면 원격파일은 newer 로 간주된다. 그밖의 경우에 이 지령은 get 와 동일하다.

nlist [remote-directory] [local-file]

원격등록부의 선택된 목록을 국부파일에 쓴다. 만일 원격등록부가 규정되지 않으면 현재 작업등록부를 리용한다. 만일 호상작용예고가 on 이면 ftp 는 마지막인수가 nlist 출력의 목표국부파일인가를 확인하는 통보를 사용자에게 내보낸다.

nmap [inpattern outputtern]

파일이름변환도구를 설정하거나 설정하지 않는다. 만일 인수들이 규정되지 않으면 파일이름변환도구는 설정되지 않는다. 인수들이 규정되면 원격파일이름은 mput 지령과 규정된 원격목적지가 없이 출구된 지령 put 을 수행할 때 변환된다. 이 지령은 보통 비 UNIX 원격콤퓨터를 다른 파일이름변환체계와 접속시킬 때 편리하다. 변환은 inpattern 과 outpattern 으로 설정되는 패턴에 따라 진행된다. inpattern 은 들어 오는 파일이름이 형타이다. (ntrans 와 case 설정에 따라 미리 처리될수있다.) 변수의 형타는 다른 inpattern 에 $1, $2, . . . , $9 을 포함시킴으로써 진행한다. $기호의 이 전문적취급을 막기 위하여 \을 리용한다. 모든 다른 기호들은 보통 기호상수로 취급되고 nmap 의 inpattern 변수들의 값의 결정에 리용한다. 실례로 inpottern 에 $1, $2 가 주어 지고 원격파일이름 mydata.data 가 주어 지면 $1 은 mydata 값을 가지고 $2 는 data 값을 가진다. outpattern 은 변환시킨 결과의 파일이름을 결정한다. $1, $2, …$9 는 inpattern 형타로부터 임의의 결과값으로 교체된다. $0 은 원래 파일이름으로 교체된다. 추가적으로 seq1 이 빈행이 아니면 [seq1, seq2]는 seq1 로 교체된다. 다른 경우에는 seq2 로 교체된다. 실례로 지령 nmap $1.$2.$3 [$1, ＄ 2].[$2, file]은 입력파일이름 myfile.data 와 myfile.data.old 에 대하여 출력파일이름 myfile.data, 입력파일이름 filename 에 대하여 myfile.file 을, 입력파일이름 .myfile 에 대하여 myfile.myfile 을 생성한다. 공백들은 다음 실례와 같이 outpattern 에 포함될수 있다.

실례 : nmap $1 | sed "s/ *$//" > $

$, [,], , 기호들의 전문적취급을 막기 위하여 \기호를 리용한다.

ntrons [inchars [outchars]]

파일이름기호의 변환수단의 설정과 비설정을 한다. 만일 인수들이 규정되지 않으면 파일이름기호의 변환수단은 설치되지 않는다. 만일 인수들이 규정되면 원격파일이름에서 기호들은 규정된 원격목표의 파일이름을 규정함이 없이 발생된 지령 mput 와 지령 put 를 수행할 때 변환된다. 만일 인수들이 규정되면 국부적파일이름에서 기호들은 규정된 국부목표의 파일이름이 없이 리용된 지령 nget 와 지령 get 를 수행할 때 변환된다. 이 지령은 보통 비 UNIX 원격콤퓨터를 다른 파일이름변환체계와 접속시킬 때 편리하다. inchars 에서 기호들에 들어 맞는 파일이름안의 기호들은 outchars 에 있는 대응하는 기호로 교체된다. 만일 inchars 에서 기호들의 위치가 outchars 의 길이보다 더 길다면 그 기호들은 파일이름으로부터 제거된다.

open server_host [port_number]

port_number 를 리용하여 server_host 와 접속을 설정한다. 만일 자동가입이 가능하면 ftp 는 봉사기호스트에 가입을 시작한다.

prompt 대화방식을 고정한다. 기정으로 ftp 는 여러개의 파일지령들을 수행할 때 매 출력파일에 반응이 있거나 없다는것을 사용자에게 통보한다. 만일 통보가 불가능하면 ftp 는 모든 규정된 파일들에 대하여 그 지령을 수행한다.

proxy ftp_command
2 차조종접속에서 지령 ftp 를 수행한다. 이 지령은 두개 봉사기들사이에 파일전송을 위한 두개의 원격 FTP 봉사기들과의 동시접속을 허용한다. 첫 지령 proxy 는 2 차조종접속을 설정하기 위하여 열기를 한다. 2 차접속에서 수행할수 있는 다른 FTP 지령들을 보기 위하여 지령 proxy ?를 입력한다. 다음의 지령들은 proxy 를 시작할 때 다르게 행동한다. open 은 자동가입의 처리를 할 때 새 마크로를 정의하지 않는다. close 는 존재하는 마크로정의를 지우지 않는다. get 와 mget 는 1 차조종접속에서의 호스트로부터 2 차조종접속에서의 호스트에로 파일들을 전송한다. put, mput, oppernd 는 2 차조종접속의 호스트로부터 1 차조종접속의 호스트에로 파일들을 전송한다. 세번째 부분파일은 2 차조종접속에서 봉사기에 의하여 FTP 규약 PASV 지령의 종속성을 전송한다.

put local_file [remote_file]
국부파일을 원격파일에 복사한다. 만일 원격파일이 규정되지 않으면 ftp 는 ntrans 혹은 nmap 설정에 따라 처리되는 국부파일이름을 원격파일이름으로 한다.

pwd 원격작업등록부의 이름을 stdout 에 쓴다.

quit bye 와 동일하다.

quote argments
봉사기호스트에 축차적으로 인수들을 보낸다(ftd(1M)을 참고).

recv remote-file [local-file]
get 와 동일하다.

reget remote-file [local-file]
reget 는 get 와 같이 류사하나 국부파일이 존재하고 원격파일보다 작다면 국부파일은 원격파일의 부분적으로 전송된 복사로 되고 전송은 실패점에서부터 계속되는것만 다르다. 이 지령은 접속이 끊어 지는 경향이 있는 망에서 대단히 큰 파일을 전송할 때 편리하다.

rhelp [command-name]
봉사기호스트로부터 방조를 요청한다. 만일 지령이름이 규정되면

봉사기에 그것을 제공한다(ftpd(1M)을 참고).

rstatus [file-name]
인수가 없을 때 원격기계의 상태를 보여 준다. 만일 파일이름이 규정되면 원격기계에서 파일이름의 상태를 보여 준다.

rename remote-from remote-to
파일 혹은 등록부로 될수 있는 remote-from 을 remote-to 로 이름을 바꾼다.

reset 반응대기렬을 지운다. 이 지령은 원격 FTP 봉사기로 command/reply 렬을 재동기화한다. 재동기화는 원격봉사기에 의하여 FTP 규약에 대한 침해가 있을 때 필요할수 있다.

restart marker
지적된 marker 로 직접 뒤따르는 get 혹은 put 를 다시 시작한다. UNIX 체계들에서 marker 는 보통 파일에서 한바이트만큼 차이난다.

rmdir remote-directory
원격등록부를 제거한다. 원격등록부는 빈 등록부로 되여야 한다.

runique 유일한 파일이름으로 국부적체계에서 파일들의 기억을 고정한다. 만일 파일이 이미 지령 get 혹은 mget 에 의해서 목표국부파일이름과 같은 이름이 존재한다면 .1 이 이름에 추가된다. 만일 결과이름이 다른 존재하는 파일에 들어 맞추어 지면 .2 가 원래 이름에 추가된다. 만일 이 과정이 .99 까지 계속된다면 오유통보가 인쇄된다. 그리고 전송은 하지 않는다. ftp 는 유일한 파일이름을 통보한다. 쉘지령으로부터 발생된 국부파일들에서의 유일성은 작용하지 않는다. 기정값은 off 이다.

send local_file [remote_file]
put 와 동일하다.

sendport
지령 port 의 리용을 고정한다. 기정으로 ftp 는 매 자료전송에 대하여 접속을 설정할 때 지령 port 를 리용하려고 한다. 만일 지령 port 가 실패하면 ftp 는 기정의 자료포구를 리용한다. 지령 port 를 리용할수 없을 때 ftp 는 매 자료전송에서 지령 port 를 접수하지 않는다. 이것은 지령 port 를 무시하는 일정한 FTP 실현에서 유용하지만(부정확하게)접속되였다는것을 지적한다(ftpd(1M)을 참고). sendport 를 off 로 하면 지령수행을 지연시킬수 있다.

site argments
지령 site 와 같이 봉사기호스트에 축차적으로 인수를 보낸다 (ftpd(1M)을 참고).

size remote_file

원격파일의 크기를 보여 준다.

status ftp의 현재 상태를 보여 준다.

struct [struct-name]

FTP의 파일전송구조를 struct-name로 설정한다. 제공되는 구조는 파일구조만이다.

sunique 유일한 파일이름으로 원격기계에서 파일들의 기억시키기를 고정한다. 원격봉사기는 유일한 이름을 통보한다. 기정으로 sunique는 off이다.

system 원격기계에서 수행하는 조작체계의 종류을 보여 준다.

tenex FTP 파일전송형을 tenex로 설정한다.

type [type_name]

FTP 파일전송형을 type_name으로 설정한다. 만일 type_name이 규정되지 않으면 현재 형을 stdout에 써넣는다. ASCII, binary, tenex는 현재 제공되는 형들이다.

umask [newmask]

원격봉사기에서 기정의 마크로를 newmask로 설정한다. 만일 망마스크가 생략되면 현재 마스크가 인쇄된다.

user user-name [password] [account]

사용자는 이미 열려 져 있어야 하는 현재 접속에서 봉사기호스트에 가입한다. 사용자의 국부홈등록부에서 .netrc 파일은 사용자이름, 통과암호, 선택적으로 계산서를 제공한다(netrc[4]를 참고). 그밖의 경우에 ftp는 사용자에게 이 통보를 내보낸다. HP-UX FTP 봉사기는 계산서를 요구하지 않는다. 안전을 담보하기 위하여 ftp는 항상 통과암호를 요구한다. 통과암호를 가지지 않는 원격콤퓨터들에는 가입할수 없다.

verbose

복잡한 출력방식을 고정한다. 만일 복잡한 출력가 가능하면 ftp는 봉사기호스트로부터 반응을 표시하고 파일전송이 완성될 때 전송의 효과에 무관계하게 통계자료를 통보한다.

? [command]

지령 help와 동일하다. 규정된 지령의 방조정보를 인쇄한다.

파일전송의 강제정지

파일전송을 강제로 끝내기 위하여서는 말단중단건(보통 ctrl-c)을 리용한다. 이때 전송은 즉시에 중지된다. ftp는 원격봉사기에 FTP 규약의 지령 ABOR를 처음으로 보냄으로써 들어 오는 전송을 정지시킨다. 이 지령이 도달되는 속도는 원격봉사기의 지령 ABOR의 처리의 제공에 따라 다르다. 만일 원격봉사기가 지령

ABOR 를 제공하지 않는다면 원격봉사기가 요청된 파일의 보내기를 완성할 때까지 ftp> 통보는 나타나지 않는다.
ftp 가 원격봉사기로부터 응답을 대기할 동안 말단중단렬은 무시된다. 이 방식에서 응답이 오래 걸리면 우에서 서술된 지령 ABOR 처리로부터의 결과로 될수 있거나 FTP 규약의 침해를 포함하여 원격봉사기에 의한 돌발적인 행동으로부터의 결과로 될수 있다. 만일 응답이 돌발적인 원격봉사기의 행동으로부터의 결과라면 국부적 ftp 프로그람은 수동적으로 끝내야 한다.

습관적인 파일이름달기

ftp 지령에서 인수들로서 규정된 파일들은 다음 규칙에 따라 처리된다.

- 만일 -가 규정되면 ftp 는 표준입력장치(읽기용) 혹은 표준출력장치(쓰기용)를 리용한다.
- 만일 파일이름의 첫 기호가 ｜이면 ftp 는 나머지 인수를 쉘지령으로 해석한다. ftp 는 주어 진 인수와 popen( )으로 쉘을 기동시키고 표준출력(표준입력)장치로부터 읽는다. (쓴다.) 만일 쉘지령이 공백을 포함하면 인수들은 " "에 포함되여야 한다. 이 기능의 간단한 실례는 "| dir. | more"이다.
- 만일 제거할수 있으면 ftp 는 C 쉘에서 리용하는 규칙에 따라 국부파일이름을 확장한다(지령 glob 를 참고). 만일 지령 ftp 가 단일한 국부파일(실례로 put 를 요구한다면 제거연산으로 생성되는 첫 파일이름만이 리용된다.
- 국부적파일이름을 규정하지 않은 지령 mget 와 지령 get 에 대하여 국부파일이름은 원격파일이름과 같은 이름을 리용한다. 그것은 case, ntrans, nmap 설정으로 변할수 있다. 결과 파일이름은 runique 가 on 이면 변경될수 있다.
- 원격파일이름을 규정하지 않으면 지령 mput 와 지령 put 에 대하여 원격파일이름은 국부파일이름과 같은 이름을 리용한다. 이 이름은 ntrans 나 nmap 설정으로 변경될수 있다. 결과 파일이름은 sunique 가 on 이면 원격봉사기에 의하여 후에 변경될수 있다.

경고

많은 지령의 정확한 수행은 원격봉사기의 고유한 행동에 의존한다.

저자

ftp 는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

관련항목

csh(1), rcp(1), ftpd(1M), netrc(4), ftpusers(4), hosts(4)

ifconfig

ifconfig － 망대면부의 파라메터들을 표시하거나 모형화한다.

---

ifconfig(1M)

이름

ifconfig － 망대면부의 파라메터들을 모형화한다.

형식

ifconfig interface address_family [adress [dest_address]] [parameters]
ifconfig interface [address_family]

해설

지령 ifconfig 의 첫번째 형식은 망대면부의 주소를 부과하거나 망대면부의 파라메터들을 모형화한다. ifconfig 는 매 대면부의 망주소를 정의하기 위하여 시동할 때 리용되여야 한다. 그것은 또한 대면부의 주소 혹은 다른 연산파라메터들의 재정의에 리용될수 있다.
주소족이 없는 이 지령의 두번째 형식은 대면부의 현재 모형을 표시한다. 만일 주소족이 규정된다면 ifconfig 는 그 주소족에서 상세한 규정만을 통보한다.
적당한 특정권을 가진 사용자만이 망대면부의 모형을 변경시킬수 있다. 모든 사용자들은 이 지령의 두번째 형식의 지령을 수행시킬수 있다.

인수들

ifconfig 는 다음의 인수들을 리용한다.

address 호스트이름자료기지에 제출된 호스트이름 혹은 인터네트표준점표시에서 표시된 DARPA 인터네트주소이다(hosts(4) 혹은 inet(3N)을 참고). 호스트번호는 10MB 리용/두번째 Ethernet 대면부(하드웨어의 물리적주소를 하는)와 첫번째 대면부와는 다른 대면부에서 생략될수 있다.

address_family
이름화도식이 기초하고 있는 규약의 이름이다. 대면부는 매개가 개별적이름도식을 요구하는 다른 규약에서 전송받을수 있다. 그러므로 그것은 지령행의 나머지 파라메터들의 해석에 리용될수 있는 주소족을 규정하는데 필요하다. 현재 제공된 주소족은 inet 이다. (DARPA 는 인터네트족이다.)

dest_address

목적지체계의 주소이다. 호스트이름자료기지에서 제출된 호스트이름 혹은 인터네트표준점표시에서 반영된 DARPA 인터네트주소로 구성된다.

interface lan0 과 같은 형식이름단위의 행이다(부분항목에서 번호화된 LAN 기판을 참고).

parameters

다음의 연산파라메터들중 한개 혹은 여러개를 리용할수 있다.

up 대면부를 up 으로 표시한다. ifconfig 를 down 한후에도 대면부는 리용할수 있다. 대면부에 주소를 설정할 때 자동적으로 나타난다. 이 기발의 설정은 하드웨어가 down 되였을 때에만 작용하지 않는다.

down 대면부를 down 으로 표시한다. 대면부가 down 으로 표시될 때 체계는 그 대면부를 통하여 통보를 전송하려고 하지 않는다. 만일 가능하면 대면부는 접수도 할수 없게 재설정할수 있다. 이 행동은 대면부를 리용하여 경로를 자동적으로 리용하지 못하게 한다.

broadcust (inet 에서만)망에서의 **방송주소**(broadcust address)를 규정한다. 이 기정의 방송주소는 호스트부분이 전부 1 로 된 주소이다.

debug 장치조종자에 의존하는 정비검사코드를 가능하게 한다. 이것은 실지로 나머지 콘솔오유를 일시적으로 가능하게 한다.

-debug 장치조종자에 의존하는 정비검사코드를 하지 못하게 한다.

ipdst (NS 만)이것은 원격망을 위하여 제한된 NS 파케트를 통합한 IP 파케트를 접수하게 하는 인터네트호스트를 규정하는데 리용된다. 이 경우에 나타나는 점으로부터 점까지의 련결은 조성되며 규정된 주소는 NS 주소와 목적지인 망으로 취한다.

metric n n 으로 대면부에서 경로화의 거리를 설정한다. 기정으로는 0 이다. 경로화거리는 경로화규약에 의하여 리용된다(gated(1M)을 참고). 더 큰 거리는 리용하기 좋은 더 작은 경로선택에 리용된다. 거리는 목적지인 망 혹은 호스트까지 추가적인 걸음으로 계산된다.

netmask mask

(inet 망)부분망으로 망의 분할 혹은 상급망으로 망의 결합을 위하여 예정된 주소가 얼마나 많은가를 규정한다. 마스크는 점표시의 인터네트주소 혹은 망의 주소표에서 목록화된 의사

망이름으로서 0x 가 앞에 놓이는 한개의 16 진수로 규정될수 있다(network(4)를 참고). 망을 부분망으로 분할하기 위하여 마스크는 국부적주소의 망부분과 주소의 호스트마당으로 될 부분망의 부분을 포함해야 한다. 마스크는 망과 부분망의 부분에 리용될 32bit 주소에서 비트위치들에 1 을, 호스트부분에 0 을 포함해야 한다. 마스크에서 1 은 32bit 마당에서 가장 왼쪽의 비트위치로부터 시작하여 련속적으로 놓여야 한다. 마스크는 적어도 표준망부분을 포함해야 하며 부분망마당은 망부분에 련달아 있어야 한다. 부분망마당은 적어도 2bit 는 포함해야 한다. 주소와 마스크사이에 자리별론리적연산을 수행한 후에 부분망부분은 모두 0 혹은 모두 1 들만 포함해서는 안된다. 망을 상급망으로 통합할 때 마스크는 망부분만 포함하여야 하며 32bit 마당의 가장 왼쪽 비트로부터 시작하여 련속적으로 1 을 포함하여야 한다.

trailers 이 기발은 보낼 때 련결차의 형태로 련결하여 정보를 통합할 것을 요청한다. 만일 망대면부가 련결차들을 봉사한다면 가능할 때 체계는 수신자에 의하여 수행되는 기억사이의 복사처리의 회수를 최소화하는 방법으로 통보의 출력을 통합한다. 주소해결의 규약을 봉사하는 망에서 이 신호기는 체계가 이 호스트에 보낼 때 다른 체계가 련결차를 리용하도록 요청한다. 류사하게 련결차통합은 이런 요청을 하는 다른 호스트에 보낼수 있다. 현재 인터네트규약에서만 리용된다(경고부분을 참고).

- trailers 련결차련결수준의 통합을 리용하지 못하게 한다(기정적).

LAN 기판번호화

LAN 기판과 관련된 대면부의 이름은 lan 이고 그 단위번호는 다음과 같이 결정된다. 체계에 처음 설정된 LAN 기판은 주어 진 대면부단위번호 0, 다음 설정된 LAN 기판은 대면부단위번호 1 등으로 설정된다. 같은 시간에 설정된 두개이상의 LAN 기판이 있다면 대면부단위번호는 **배경판**(Backplane)에서 기판위치에 따라 부과된다. 배경판에서 처음으로 나타난 LAN 기판은 대면부단위번호 N 으로 주어 진다. 배경판에서 LAN 기판은 수 N+1 로 주어 진다.

지령 lanscan 은 LAN 기판과 관련된 매 대면부의 이름과 단위번호를 표시하는데 리용될수 있다.

상급망들

상급망은 더 작은 망들을 포함하는 모임이다. 상급망화는 상급망보다 더 작은 망의 모임을 상급망으로 통합하기 위한 망리용의 기술이다. 이 기술은 특히 클

라스 C 의 방식에서 유용하다. 클라스 C 의 망은 254 개의 호스트들만 가진다. 이것은 일부 회사들에서 너무 큰 제한으로 될수 있다. 이런 회사들에 대하여 망의 부분망을 포함하는 한 망을 상급망으로 형성하면서 이 클라스 C 의 망에서 호스트들로 적용될수 있다. 이 상급망의 망마스크는 지령 ifconfig 의 리용으로 그 상급망에 접속된 대면부들에 적용되여야 한다. 실례로 어떤 호스트를 클라스 C 의 상급망에 접속하기 위하여 대면부를 모형화할수 있다. 그리고 192.6.1.1 의 IP 주소모형화는 192.6 과 그 대면부에 255.255.0.0 의 망으로 모형화할수 있다.

진단

통보는 규정된 대면부가 존재하지 않으면 지적하고 요구된 주소가 알려 지지 않거나 사용자가 특정권이 없고 대면부의 모형화를 시도하려고 할 때 지적되게 된다.

경고

현재 모든 HP9000 체계는 련결차파케트를 받을수 있으나 그것을 보내지는 못한다. 련결차기발설정은 작용하지 않는다.

관련항목

netstat(1), lanconfig(1M), lanscan(1M), hosts(4), routing(n)

netstat

netstat - 망과 관련된 통계자료를 표시한다.

---

netstat(1)

이름

netstat - 망의 상태를 표시한다.

형식

netstat [-aAn] [-f address_family] [system [core]]
netstat [-mMnrsv] [-f address_family] [-p protocol] [system [core]]
netstat [-gin [-I interface] [interval] [system [core]]

해설

netstat 는 여러가지 망과 관련된 자료구조의 내용은 물론 망대면부와 규약에 대한 통계자료를 표시한다. 출력형식은 지적된 선택항목에 따라 변한다. 일부 선택항목들은 다른 선택항목들과 결합하여 리용될 때 무시된다.

일반적으로 지령 netstat 은 우에서 보여 준 세 형식들중의 하나를 취한다.

- 첫번째 형식의 지령은 매 규약의 **능동소케트**(Active Socket)의 목록을 표시한다.
- 두번째 형식은 지적된 선택항목에 따라 다른 망자료구조들중의 한가지 내용을 표시한다.
- 세번째 형식은 매 망대면부의 모형정보를 표시한다. 그것은 또한 선택적으로 매 구간에서 갱신되며 초단위로 측정된 모형화된 망대면부에서 망의 통신자료도 표시한다.

이 지령은 다음과 같은 선택항목들을 가진다.

-a 봉사기프로쎄스들에 의하여 리용된 **피동소케트**(Passive Socket)를 표함하여 모든 소케트들의 상태를 보여 준다. netstat 가 (-A 와 -n 을 제외하고)임의의 선택항목없이 리용될 때 능동적소케트만 보여 준다. 이 선택항목들은 X.25 프로그람적접근소케트의 상태를 보여 주지 않는다. 이 선택항목은 만일 선택항목 -g, -i, -I, -m, -M, -p, -r, -s 혹은 interval 이 규정되면 무시된다.

-A 소케트와 관련된 규약조종블로크의 주소를 보여 준다. 이 선택항목은 장치검사에 리용된다. 그것은 X.25 프로그람적접근조종블로크를 보여 주지 않는다. 이 선택항목은 만일 선택항목 -g, -i, -I, -m, -M, -p, -r, -s 혹은 interval 이 규정되면 무시된다.

-f address-family
규정된 주소족에 대한 통계자료 혹은 주소조종블로크만 보여 준다. 다음과 같은 주소족을 리용한다. 즉 AF_INET 에 대하여서는 inet, AF_UNIX 에 대하여서는 unix 를 리용한다. 이 선택항목들을 선택항목 -a, -A, -s 들과 함께 적용한다.

-g 망대면부에 대한 다중공개된 정보를 보여 준다. 이 선택항목은 주소족 AF_INET 만을 식별한다. 이 선택항목은 두 종류의 정보를 표시하기 위하여 선택항목 -i 와 결합될수 있다. 이 선택항목은 선택항목 -m, -M, -p 가 규정될 때 무시된다.

-i 망대면부의 상태를 보여 준다. 체계에 정적으로 모형화될수 있으나 시동될 때 배치되지 않은 대면부들은 보여 주지 않는다. 이 선택항목은 선택항목 -m, -M, -p 가 규정되면 무시된다.

-I Interface

규정된 대면부에 대하여서만 정보를 보여 준다. 이 선택항목은 선택항목 -g, -i와 함께 적용된다.

-m 망기억관리루틴으로 기록된 통계자료를 표시한다. 만일 이 선택항목이 규정되면 모든 다른 선택항목들은 무시된다.

-M 다중으로 공개된 경로화표를 보여 준다. 선택항목 -s가 선택항목 -M과 함께 리용될 때 netstat는 다중으로 공개된 경로화의 통계자료를 표시한다. 이 선택항목은 선택항목 -m이나 선택항목 -p가 규정되면 무시된다.

-n 수값으로서 망주소들을 보여 준다. 표준적으로 netstat는 주소들을 해석하고 그것을 기호적으로 표시하려고 한다. 이 선택항목은 선택항목 -a, -A, -i, -r, -v들과 함께 리용된다.

-p protocol

규정된 규약에 따라 통계자료를 표시한다. 다음의 규약 tcp, udp, ip, icmp, igmp, arp, probe들을 식별한다. 이 선택항목들은 선택항목 -m이 규정되면 무시된다.

-r 경로화표를 보여 준다. 선택항목 -v가 선택항목 -r와 함께 리용될 때 netstat는 경로의 등록항목들에서 망마스크들을 표시한다. 선택항목 -s가 선택항목 -r와 같이 리용될 때 netstat는 실지로 경로화통계자료를 표시한다. 이 선택항목은 선택항목 -g, -n, -M, -i, -I, -p, interval이 규정되면 무시된다.

-s 모든 규약들에 대한 통계자료를 보여 준다. 이 선택항목이 선택항목 -r와 함께 리용될 때 netstat는 실지로 경로화의 통계자료를 표시한다. 이 선택항목이 선택항목 -M과 함께 리용될 때 netstat는 실지로 다중으로 공개된 경로화의 통계자료를 표시한다. 이 선택항목은 선택항목 -g, -i, -I, -m, -p, interval과 함께 규정되면 무시된다.

-v 추가적경로화의 정보를 보여 준다. 선택항목 -v가 선택항목 -r와 함께 리용될 때 netstat는 경로의 등록항목에서 **망마스크**(Network Mask)들도 표시한다. 이 선택항목은 선택항목 -r와 함께 리용된다.

인수들인 system과 core는 기정으로 /stand/vmunix와 /dev/kmem의 대입을 허용한다.

만일 선택항목들이 없거나 선택항목 -A 나 -n 만 규정되면 netstat 는 응용프로그람 소케트의 상태만을 표시한다. 능동및 피동소케트들의 상태표시는 국부 및 원격 주소들, 보내거나 받은 순서의 크기(바이트로), 규약, 규약의 내부상태를 보여 준다. 주소형식들은 host.port 형식이거나 만일 소케트주소들의 호스트부분이 0 이라면 net work.port 형식이다. 알려 질 때 호스트와 망주소들은 각각 gethost byname()과 getnetbyname()을 리용하여 기호적으로 표시된다(gethost byname(3N)과 getnetbyname(3N)을 참고). 만일 주소의 기호적이름이 알려 지지 않거나 선택항목 -n 이 규정되면 주소는 주소족에 따라 수값으로 표시된다. 인터네트의 점형식에 무관계하게 더 많은 정보를 알기 위하여서는 inet(3n)을 참고하면 된다. 규정되지 않았거나 그룹기호로 표시된 주소와 포구들은 한개의 *로 나타난다.

대면부는 전송된 파케트들, 오유들, 충돌들에 무관계하게 축적된 통계자료의 표를 제공한다. 대면부의 망주소와 최대전송단위(MTU)를 표시한다. interval 인수가 규정될 때 netstat 는 망대면부에 관계된 통계자료에서 수행한 계수기를 표시한다. 이 표시는 1 차대면부의 렬(첫 대면부는 자동모형화일 때 찾아 진다. 과 모든 대면부들에 대한 합계정보의 렬로 구성된다. 1 차대면부를 다른 대면부로 교체하려면 선택항목 -I 를 리용하여야 한다. 매개 정보화면의 첫 행은 체계가 마지막으로 재기동된 후에 개요로 표시된다. 출력된 순차적행들은 선행한 구간에서 합계된 값을 보여 준다.
경로화표의 표시는 리용할수 있는 경로와 그 상태를 지적한다. 매 경로는 목적지호스트 혹은 망, 망마스크, 앞으로 파케트들에서 리용될 관문으로 구성된다. 기발마당은 경로가 우로 (U), 경로가 관문 (G), 경로가 호스트 혹은 망경로(H 가 있거나 없이), 경로가 재방향 혹은 경로 MTU 의 등록부에 의하여 동적으로 (D)로 조성되는가, 관문경로가 변경되는가 (M), 시간적으로 ARP 반응의 부족으로 표식 (?)이 나타나는가를 보여 준다.
마당 Netmask 은 앞으로 IP 파케트의 목적지 IP 주소에 적용될 마스크를 보여 준다. 결과는 경로등록항목에서 목적지주소와 비교될것이다. 만일 그것들이 같다면 경로는 이 IP 파케트의 경로화를 위한 **후보자**(Candidate)들중의 하나로 된다. 만일 후보자경로가 여러개 있다면 가장 긴 망마스크마당(가장 왼쪽 비트위치로부터 시작하여 련속적으로 1 들이 놓여 있는 마당)으로 경로가 선택될것이다.

마당 Geteway 은 목적지에 도달하기 위한 중간관문의 주소들을 보여 준다. 그것은 만일 목적지가 직접 접속된 망에 있다면 나가는 대면부의 주소로 될수 있다. 마당 refs 은 경로의 능동적리용의 현재 개수를 보여 준다. 접속방향규약은 접속규약의 특이한 통보를 보낼 때 경로를 표준적으로 얻도록 하면서 접속의 지속을 위한 단일한 경로를 유지하게 한다. 리용마당은 경로를 리용하여 보낸 파케트의 개수를 보여 준다. 대면부마당은 어떤 망대면부가 경로에 리용되는가를 동일화한다.

마당 Pmtu 와 PmtuTime 들은 호스트경로에만 적용된다. 망과 기정의 경로에 대하여 pmtu 마당은 그 경로에 리용된 망대면부의 MTU 와 같은것이다. 만일 경로가 정적인 PMTU 의 값으로 조성된다면 대응하는 마당 PmtuTime 은 단어 perm 을 포함하고 PMTU 값은 항상 대면부의 MTU 를 무시한다. 만일 경로가 동적으로 조성된다면 (기발마당에서 D) 대응되는 마당 PmtuTime 에서 값은 PMTU 가 없어지기전에 남아 있는 분단위로 표시된 수값이다. PMTU 가 제거될 때 변경된 경우에 체계는 경로를 위한 현재 PMTU 를 다시 발견하게 된다. 마당 PmtuTime 은 PMTU 가 대면부의 MTU 와 동일시될 때 공백으로 된다. 마당 Pmtu 에서 *는 사용자가 경로에 대한 PMTU 를 제거하지 못하였다는것을 보여 준다.

종속성

X.25 :
선택항목 -A 와 -a 들은 X.25 프로그람적접근정보를 목록화하지 않는다.

저자

netstat 는 캘리포니아종합대학, 버클리에 의해서 개발되였다.

관련항목

hosts(4), network(4), gethostbyname(3N), getnetbyname(3N), protocols(4), route(1M), services(4)

ping

ping – 망우에서 정보를 보내고 응답을 받는다.

---

ping(1M)

이름

ping – 망호스트에 ICMP 반사요청파케트를 보낸다.

형식

ping [-oprv] [-i address] [-t ttl] host [-n count]
ping [-oprv] [-i address] [-t ttl] host packet_size [[-n] count]

해설

지령 ping 은 ICMP **반사요청**(Echo Request) (ECHO_REQUEST)파케트를 초당 한 번씩 호스트에 보낸다. ICMP 반사요청파케트를 통하여 거꾸로 반사되는 매 파케트는 왕복전송시간을 포함하여 표준출력으로 씌여 진다.

ICMP 반사요청자료기입(pings)은 IP 와 ICMP 머리부, 그뒤에 구조 timeval 과 파케트채우기에 리용될 바이트의 임의의 개수를 가진다. 이것은 파케트의 크기선택항목을 리용하여 변경시킬수 있는데 기정자료기입의 길이는 64byte 이다.

선택항목들

ping 은 다음의 선택항목들과 파라메터들을 가진다.

-i address 호스트가 다중공개주소라면 다중공개자료기입을 대면부로부터 주소가 "."으로 규정된 국부 IP 주소로 보낸다. 선택항목 -i 가 규정되지 않는다면 다중공개자료기입을 경로모형화로 결정되는 기정대면부로부터 보낸다.

-o 지령이 종결될 때 취한 경로들을 개괄하면서 나가는 파케트에 IP 기록경로선택항목을 기입한다. 만일 이 경로에서 일부 호스트들이 IP 기록의 경로선택항목을 실현하지 않는다면 왕복전송경로를 얻는것은 불가능하다. 최대로 9 개의 인터네트주소들이 선택항목 IP 령역에 최대길이로 기록될수 있다.

-p ICMP 의 통보 "Datagram Too Big"를 관문으로부터 받을 때 새로운 경로 MTU 정보는 표시된다. 선택항목 -p 가 선택항목 -v 는 큰 파케트크기와 련결될 때 리용되여야 한다.

-r 표준경로표를 무시하고 도달된 망의 호스트에 직접 보낸다. 만일 호스트가 직접 접속된 망에 없다면 오유는 귀환된다. 이 선택항목은 대면부가 geted 로 제거된것과 같이 경로화되지 않는 대면부를 통하여 국부체계의 ping 에 리용될수 있다(geted(1M)을 참고).

-t ttl 호스트가 다중공개주소이라면 다중공개자료기입에서 마당 time_to_live 을 tt1 로 설정한다. 자료기입이 전달되는 과정을 통하여 외부적체계에 대한 최대개수의 규정으로 다중공개자료기입의 범위를 조종한다.

만일 tt1 이 0 이면 자료기입은 국부체계에서만 리용된다. 만일 tt1 이 1 이면 자료기입은 선택항목 -i 로 규정된 대면부와 직접 접속된 망에서 어느한 대면부를 가지는 체계에서만 리용된다. 만일 tt1 이 2 이면 자료기입은 대부분 한개 다중공개경로전달자에 의하여 전달될수 있다. 그리고 범위는 0 부터~255 까지이다. 기정값은 1 이다.

-v 불필요한 출력이다. 받은 **반사응답**(Echo Response)파는 다른 ICMP 파케트들을 보여 준다.

host ICMP 반사요청을 보낸 목적지이다. 호스트는 호스트이름 혹은 인터네트주소이다. 호스트로 규정된 모든 기호적이름들은 gethostbyname( )으로 알수 있다. 만일 호스트가 인터네트주소이면 그것은 "."으로 되여 있어야 한다.

만일 체계가 응답하지 않으면 경로는 국부체계 혹은 원격체계에서 혹은 중간관문에서 부정확하게 모형화되였으며 어떤 다른 망의 실패로 부정확하게 모형화되였다는것을 의미한다. 표준적으로 호스트는 국부망대면부 및 원격망대면부에 부과된 주소이다.

만일 호스트가 방송된 주소이면 방송을 받은 모든 체계들은 응답할것이다. 표준적으로 ICMP 반사응답을 보낸 국부대면부와 같이 망에서는 망대면부를 가지는 체계들만 있다.

만일 호스트가 다중공개주소이면 다중공개그룹을 련결시킨 체계들만 응답할것이다. 호스트는 선택항목 -t 가 규정되였다면 체계를 구별할수 있다. 선택항목 -i 로 규정된 대면부와 직접 접속된 망에 다중공개경로의 해결자가 있다.

packet_size
바이트로 전송된 파케트의 크기이다. 표준으로(packet_size 가 규정되지 않은) 전송된 파케트의 크기는 64byte 이다. 파케트크기로 허용되는 최소값은 8byte 이고 최대로 4095byte 이다. 만일 파케트의 크기가 16byte 보다 작다면 반복통보를 위한 충분한 령역은 없게 된다. 이 경우에 왕복전송시간은 표시되지 않는다.

count 통보의 보내기와 응답받기인 ping 의 파케트의 개수는 종결전에 전송된것이다. 범위는 0−2147483647 이다. 기정값은 0 이다. 그 경우에 ping 은 중단될 때까지 파케트를 보낸다.

실패의 차단을 위하여 ping 을 리용할 때 우선 국부망대면부는 정확히 작업하는가를 검사하기 위하여 호스트의 국부주소를 규정한다. 다음에 실패의 위치를 결정하기 위하여 호스트와 관문의 주소를 더 규정한다. ping 은 초당 한개의 자료기입을 보내고 받은 매 ICMP 반사응답에 대하여 표준적으로 한개 행씩 출력한다. 만일 응답이 없다면 출력은 산생되지 않는다. 만일 선택적계수기가 주어 진다면 규정된 요청의 수만 보낸다. 왕복전송시간과 상실된 파케트에 대한 통계자료들은 계산된다. 모든 응답을 받았거나 나간 지령시간(만일 선택항목 count 가 규정되면) 혹은 만일 지령이 SIGINT 로 종결되였다면 간단한 통보들만을 표시한다. 이 지령은 망작업수행의 검사, 관리, 측정에 리용하는데 편리할것이다. 그것은 우선 망실패를 차단하기 위하여 리용된다. 망에서 과중한 적재를 피하려면 표준

처리나 자동화된 스크립트안에서 불필요하게 ping 을 리용하지 말아야 한다.

저자

ping 은 Public Domain 에서 개발되였다.

파일들

/etc/hosts

관련항목

gethostbyname(3N), inet(3N)

rcp

rcp - 한 체계로부터 다른 체계로 파일과 등록부들을 복사한다.

---

rcp(1)

이름

rcp - 원격파일을 복사한다.

형식

단일파일 복사

rcp [-p] source_file1 dest_file

다중파일 복사

rcp [-p] source_file1 [source_file2]…dest_dir

한개 혹은 여러개 등록부의 부분나무들의 복사

rcp [-p] -r source_dir1 [source_dir2]…dest_dir

파일들과 등록부의 부분나무들의 복사

rcp [-p] -r file_or_dir1 [file_or_dir2]…dest_dir

해설

지령 rcp 는 한개 혹은 그 이상의 체계들로부터 파일들, 등록부의 부분나무들, 파일과 보조등록부의 부분나무들의 결합을 다른 체계에 복사한다. 여러가지 관점에서 볼 때 이것은 지령 cp 와 류사하다(cp(1)을 참고).

rcp 를 리용하려면 복사되는 파일들에 읽기접근을 가져야 하고 등록부의 경로에서 모든 등록부들에 읽기와 찾기(수행)허락을 가져야 한다.

선택항목들와 인수들

rcp 는 다음과 같은 선택항목들과 인수들을 가진다.

source_file
source_dir
: 규정된 목적지에 복사될 국부기계 혹은 원격기계에서 존재하는 파일이름 혹은 등록부의 이름이다. 원천파일과 등록부의 이름은 다음과 같이 구성된다.

    user_name@hostname:pathname/filename

  혹은

    user_name@hostname:pathname/dirname

  파일과 등록부이름의 성분들은 아래에서 서술된다. 만일 다중으로 존재하는 파일들과 등록부부분나무들이 규정된다면 (source_file1, source_file2, …등등) 목적지는 어느한 등록부로 되여야 한다. 쉘파일이름의 확장은 국부체계와 원격체계들에서 진행된다. 다중파일들과 등록부의 부분나무들은 한개 지령으로 한개 목적지등록부에 한개 혹은 여러개 체계들로부터 복사될수 있다.

dest_file
: 목적지파일의 이름이다. 만일 호스트이름과 경로이름이 규정되지 않으면 이미 존재하는 파일은 국부체계에서 현재등록부에 dest_file 이름으로 된 파일로 복사된다. 만일 dest_file 이 이미 존재하고 쓸수 있다면 존재하는 파일은 덧쓰게 된다. 목적지파일이름은 원천파일이름과 같은 방법으로 규정할수 있는데 파일이름의 확장자들은 리용하지 말아야 한다.

dest_dir
: 목적지등록부의 이름이다. 만일 호스트이름과 경로이름이 규정되지 않으면 이미 존재하는 파일은 국부체계의 현재등록부에서 dest_dir 로 이름지어 진 등록부에 복사된다. 만일 dest_dir 가 이미 규정된 등록부의 경로에 존재하면(규정되지 않았다면 현재등록부) dest_dir 로 이름지어 진 새 등록부는 dest_dir 라는 이름을 가진 이미 존재하는 등록부에 조성된다. 목적지등록부이름은 원천등록부의 나무이름과 같이 규정될수 있는데 파일이름의 확장자는 리용하지 말아야 한다.

file_or_dir
: 만일 복사하기 위하여 파일과 등록부들의 결합이 규정되고(명백히 혹은 파일이름확장으로) 선택항목 -r 가 규정되지 않았다

면 오직 그 파일들만 복사한다. 만일 선택항목 -r 가 규정되면 이름이 file_ or_dir 으로 규정된 대응되는 모든 파일들과 등록부의 부분나무들은 복사된다.

-p	파일조성방식마스크 umask 의 현재 설정을 무시하면서 원천파일들의 변경시간과 방식(허락)을 유지한다. 만일 선택항목 -p 가 규정되지 않으면 rcp 는 그것이 이미 존재할 때 dest_file 의 허락과 소유자는 보존된다. 다른 경우에 rcp 는 목적지호스트에서 umask 로 변경된 원천파일의 방식을 리용한다. 목적지파일의 변경과 접근시간은 복사가 만들어 질 때의 시간으로 설정된다.
-r	원천등록부의 이름에서 규정된 등록부의 부분나무를 귀납적으로 복사한다. 만일 임의의 등록부의 부분나무가 복사된다면 rcp 는 귀납적으로 규정된 원천등록부의 이름에 있는 매 부분나무를 등록부 dest_dir 에 복사한다. 만일 source_dir 가 같은 이름이 존재하는 등록부에 복사된다면 rcp 는 dest_dir 안에서 새 등록부 source_dir 를 조성하고 source_dir 로 규정된 부분나무는 dest_dir 혹은 source_dir 에 복사된다. 만일 dest_dir 가 존재하지 않으면 rcp 는 그것을 조성하고 source_dir 로 규정된 부분나무를 dest_dir 에 복사한다.

파일과 등록부이름의 규정

우에서 지적한바와 같이 파일과 등록부의 이름은 1 개, 2 개 혹은 4 개 성분들을 포함한다.

user_name	원격체계에서 접근하는 등록부들과 파일들에 대하여 리용된 가입자의 이름이다.
hostname	등록부들과 파일들이 있는 원격체계의 호스트들의 이름이다.
pathname	사용자의 user_name 의 가입등록부에 관한 등록부의 경로이름 혹은 절대적등록부의 경로이름이다.
filename	원천파일 혹은 등록부파일의 실제적이름이다. 파일이름확장은 원천파일이름에서 허용된다.
dirname	원천등록부 혹은 목적지등록부의 부분나무의 실제적이름이다. 파일이름확장은 원천등록부의 이름들에서 허용된다.

매 파일 혹은 등록부인수는 hostname:path 형식의 원격파일이름이거나 임의의 두

점(:)전에 빗선(/)으로 규정된 국부파일이름이다. hostname 은 공식적인 호스트이름 혹은 별명으로 될수 있다. 만일 hostname 이 ruser @ rhost 형식을 가지면 ruser 는 현재사용자이름대신에 원격적호스트에서 리용된다. 규정되지 않은 경로(hostname:)는 원격적사용자의 가입등록부를 참조한다. 만일 경로가 / 로 시작되지 않았다면 그것은 hostname 에서 원격적사용자의 가상등록부에 관하여 해석되게 된다. 그것들이 원격적으로 해석되도록 하기 위하여 원격적경로에서 쉘의 메타기호들은 거꿀빗선(\), 단일인용부호(' '), 2중인용부호(" ")로 인용될수 있다.

rcp 루틴은 통과암호를 예고하지 않는다. 현재국부사용자의 이름 혹은 ruser 로 규정된 임의의 사용자이름은 rhost 에 있어야 하며 remsh(1)과 rcmd(3)으로 원격적지령의 수행을 허용하여야 한다. remshd(1M)은 원격호스트에서 수행되여야 한다.
세번째 부분은 다음 형식으로 전송한다.

rcp ruser1 @ rhost1:path1 ruser2 @ rhost2:path2

은

remsh rhost1 -1 ruser1 rcp path1 ruser2 @ rhost2:path2

로 수행된다.

그러므로 성과적인 전송을 위하여 rhost2 에서 ruser2 는 rhost1 로부터 ruser1 에 의하여 접근이 허락되여야 한다.

경고

rcp 루틴은 원격적호스트의 .cshrc 파일에서 지령들에 의하여 발생된 임의의 출력과 혼돈할수 있다.
실례로 자체로 파일을 복사하려고 지령

rcp path 'hostname' : path

을 주면 모순되는 결과가 얻어 진다. rcp 의 현재 HP-UX 방안은 자체로 파일을 단순히 복사한다. 그러나 HP-UX 이전방안에서 rcp 의 실현은 파일을 복사하면 파일이 못쓰게 되는 때도 있다. 추가로 같은 파일은 다른 방법으로 참조될수도 있다. 실례로 **굳은련결**(Hard Link), 기호적련결, NFS 를 통하여 참조될수도 있다. rcp 가 모든 경우에 정확히 파일을 복사할것이라는것은 담보되지 않는다.

HP-UX 7.0 이전의 방안과 4.2 BSD 방안에서 rcp 의 실현은 원격적사용자들이 rhost . ruser 로 규정할것을 요구한다. 만일 세번째 부분전송에서 규정된 첫 원격적호스트가 낡은 문법형식을 리용한다면 그 지령은 목적지를 rhost1 로 해석하기 때문에 다음 형식으로 되여야 한다.

rcp ruser1 @ rhost1 : path1 rhost2 . ruser2 : path2

공통적으로 만나게 되는 문제는 두 원격파일들이 원격적사용자를 규정하는 원격목적지에 복사된다는것이다. 만일 두개의 원천원격체계인 rhost1 과 rhost2 를 가진다면 원격목적지에 대하여 이 매개가 다른 형식을 요구할 때 다음과 같은 지령으로 되여야 한다.

rcp rhost1 : path1 rhost2 : path2 rhost3 : ruser3 : path3

이것은 원천체계들중의 어느 하나의 체계에서 확실히 실패할것이다. 두개의 개별적지령으로 이런 전송을 수행해야 한다.

저자

rcp 는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

관련항목

cp(1), ftp(1), remsh(1), remshd(1M), rcmd(3), hosts(4), hosts . equiv(4)
Using Internet Services 에서 ftp 장을 참고.

rcp(1) kerberos 인증으로 인터네트봉사를 안전하게 한다. rcp(1)

이름

rcp - 원격파일복사

형식

단일파일 복사

rcp [-k realm] [-P] [-p] source_file1 dest_file

다중파일 복사

rcp [-k realm] [-P] [-p] source_file1 [source_file2]…dest_dir

한개 혹은 여러개 등록부의 부분나무들의 복사

rcp [-k realm] [-P] [-p] -r source_dir1 [source_dir2]…dest_dir

파일들과 등록부의 부분나무들의 복사

rcp [-k realm] [-P] [-p] -r file_or_dir1 [file_or_dir2]…dest_dir

해설

지령 rcp 는 한개 혹은 여러개 체계들로부터 파일들, 등록부의 부분나무들, 파일과 보조등록부의 부분나무들의 결합을 다른 체계에 복사한다. 여러가지 관점에서 볼 때 이것은 지령 cp 와 류사하다(cp(1)을 참고).
rcp 를 리용하려면 복사되는 파일들에 읽기접근을 가져야 하고 등록부의 경로에서 모든 등록부들에 읽기와 찾기(수행)허락을 가져야 한다.

Kerberos V5 망의 인증환경에서 rcp 는 원격호스트에 접속을 시도할동안 Kerberos V5 의 규약을 리용한다. 인증수단은 원격호스트에서 remshd 를 기동시킬 때 리용된 지령행선택항목들(−K, -r, -R, -k)에 의존한다. Kerberos 인증과 인증규칙은 인터네트봉사안전에 대한 기본페지의 sis(5)에 서술되여 있다.
비록 Kerberos 인증과 **권한부여**(Authorization)는 적용할수 있지만 Kerberos 수단으로 파일을 복사할 때에는 적용되지 않는다. 파일들은 망에서 명백한 본문으로 전송되게 된다.

선택항목들와 인수들

rcp 는 다음과 같은 선택항목들과 인수들을 가진다.

source_file
source_dir
: 규정된 목적지에 복사될 국부기계 혹은 원격기계에서 존재하는 파일 혹은 등록부의 이름이다. 원천파일과 등록부의 이름은 다음과 같이 구성된다.

user_name@hostname:pathname/filename

혹은

user_name@hostname:pathname/dirname

파일과 등록부이름의 성분들은 아래에서 서술된다. 만일 다중으로 존재하는 파일들과 등록부부분나무들이 규정된다면(source_file1, source_file2, …등등) 목적지는 어느한 등록부로 되여야 한다. 쉘파일이름의 확장은 국부체계와 원격체계들에서 진행된다. 다중파일들과 등록부의 부분나무들은 한개 지령으로 한개 목적지등록부에 한개 혹은 여러개 체계들로부터 복사될수 있다.

dest_file
: 목적지파일의 이름이다. 만일 호스트이름과 경로이름이 규정되지 않으면 이미 존재하는 파일은 국부체계에서 현재등록부에 dest_file 이름으로 된 파일로 복사된다. 만일 dest_file 이 이미 존재하고 쓸수 있다면 존재하는 파일은 덧쓰게 된다. 목적지파일이름은 원천파일이름과 같은 방법으로 규정할수 있는데 파일이름의 확장자들은 리용하지 말아야 한다.

dest_dir
: 목적지등록부의 이름이다. 만일 호스트이름과 경로이름이 규정되지 않으면 이미 존재하는 파일은 국부체계의 현재등록부에서 dest_dir 로 이름지어 진 등록부에 복사된다. 만일 dest_dir 가 이미 규정된 등록부의 경로에 존재하면(규정되지 않았다면

현재등록부) dest_dir 로 이름지어 진 새 등록부는 dest_dir 라는 이름을 가진 이미 존재하는 등록부에 조성된다. 목적지등록부 이름은 원천등록부의 나무이름과 같이 규정될수 있는데 파일 이름의 확장자는 리용하지 말아야 한다.

file_or_dir 만일 복사하기 위하여 파일과 등록부들의 결합이 규정되고(명백히 혹은 파일이름확장으로) 선택항목 -r 가 규정되지 않았다면 오직 그 파일들만 복사한다. 만일 선택항목 -r 가 규정되면 이름이 file_ or_dir 으로 규정된 대응되는 모든 파일들과 등록부의 부분나무들은 복사된다.

-k realm 구성파일 krb.realms 에서 규정된것처럼 원격호스트들의 기정지대를 리용할 때 규정된 realm 으로 원격호스트로부터 입장권을 얻는다.

-P Kerberos 인정을 할수 없게 한다. Kerbers V5 에 기초한 안전환경에서만 리용될수 있다. 만일 원격호스트가 비안전접근의 제출을 위하여 모형화된다면 이 선택항목들의 리용은 일반적으로 오유로 나타날것이다.

rcmd:connect: <hostname>:connection reused

더 상세한것을 알려면 remshd(1M)의 진단을 참고하여야 한다.

-p 파일조성방식마스크 umask 의 현재 설정을 무시하면서 원천파일들의 변경시간과 방식(허락)을 유지한다.
만일 선택항목 -p 가 규정되지 않으면 rcp 는 그것이 이미 존재할 때 dest_file 의 허락과 소유자는 보존된다. 다른 경우에 rcp 는 목적지호스트에서 umask 로 변경된 원천파일의 방식을 리용한다. 목적지파일의 변경과 접근시간은 복사가 얻어 질 때의 시간으로 설정된다.

-r 원천등록부의 이름에서 규정된 등록부의 부분나무를 귀납적으로 복사한다. 만일 임의의 등록부의 부분나무가 복사된다면 rcp 는 귀납적으로 규정된 원천등록부의 이름에 있는 매 부분나무를 등록부 dest_dir 에 복사한다. 만일 source_dir 가 같은 이름이 존재하는 등록부에 복사된다면 rcp 는 dest_dir 안에서 새 등록부 source_dir 를 조성하고 source_dir 로 규정된 부분나무는 dest_dir 혹은 source_dir 에 복사된다. 만일 dest_dir 가 존재하지 않으면 rcp 는 그것을 조성하고 source_dir 로 규정된 부분나

무를 dest_dir 에 복사한다.

파일과 등록부이름의 구성

우에서 지적한바와 같이 파일과 등록부의 이름은 1 개, 2 개 혹은 4 개 성분들을 포함한다.

user_name	원격체계에서 접근하는 등록부들과 파일들에 대하여 리용된 가입자의 이름이다.
hostname	등록부들과 파일들이 있는 원격체계의 호스트들의 이름이다.
pathname	사용자의 user_name 의 가입등록부에 관한 등록부의 경로이름 혹은 절대적등록부의 경로이름이다.
filename	원천파일 혹은 등록부의 실제적이름이다. 파일이름확장은 원천파일이름에서 허용된다.
dirname	원천파일 혹은 목적지등록부의 부분나무의 실제적이름이다. 파일이름확장은 원천등록부의 이름들에서 허용된다.

매 파일 혹은 등록부인수는 hostname**:**path 형식의 원격파일이름이거나 임의의 두점(**:**)전에 빗선(/)으로 규정된 국부파일이름이다. hostname 은 공식적인 호스트이름 혹은 별명으로 될수 있다. 만일 hostname 이 ruser @ rhost 형식을 가지면 ruser 는 현재사용자이름대신에 원격적호스트에서 리용된다. 규정되지 않은 경로 hostname**:**)는 원격적사용자의 가입등록부를 참조한다. 만일 경로가 / 로 시작되지 않았다면 그것은 hostname 에서 원격적사용자의 가상등록부에 관하여 해석되게 된다. 그것들이 원격적으로 해석되도록 하기 위하여 원격적경로에서 쉘의 메타기호들은 거꿀빗선(\), 단일인용부호(' '), 2 중인용부호(" ")로 인용될수 있다. rcp 는 통과암호를 예고하지 않는다. 비안전환경 혹은 습관적환경에서 만일 현재 국부사용자의 이름 혹은 임의의 사용자이름이 ruser 를 통하여 규정된다면 사용자인정은 rhost 에 존재하는가를 검사한다. Kerberos V5 망의 인정 혹은 안전환경에서 인정방법은 remshd 의 지령행의 인수들에 의존한다. 다른 경우에 remsh(1)과 rcmd(3)을 통하여 원격적지령수행은 허용되여야 하며 remshd(1M)은 원격적호스트에서 수행될수 있어야 한다.

세번째 부분은 다음 형식으로 전송한다.

rcp ruser1 @ rhost1**:**path1 ruser2 @ rhost2**:**path2

은

remsh rhost1 -1 ruser1 rcp path1 ruser2 @ rhost2:path2

로 수행된다.

그러므로 성과적인 전송을 위하여 rhost2 에서 ruser2 는 rhost1 로부터 ruser1 에 의

하여 접근이 허락되여야 한다.

경고

rcp 루틴은 원격적호스트의 .cshrc 파일에서 지령들에 의하여 발생된 임의의 출력과 혼돈할수 있다.
실례로 자체로 파일을 복사하려고 지령

rcp path 'hostname' : path

을 주면 모순되는 결과가 얻어 진다. rcp 의 현재 HP−UX 방안은 자체로 파일을 단순히 복사한다. 그러나 HP−UX 이전방안에서 rcp 의 실현은 파일을 복사하면 파일이 못쓰게 되는 때도 있다. 추가로 같은 파일은 다른 방법으로 참조될수도 있다. 실례로 굳은련결, 기호적련결, NFS 를 통하여 참조될수도 있다. rcp 가 모든 경우에 정확히 파일을 복사할것이라는것은 담보되지 않는다.
HP−UX 7.0 이전의 방안과 4.2 BSD 방안에서 rcp 의 실현은 원격적사용자들이 rhost . ruser 로 규정할것을 요구한다. 만일 세번째 부분전송에서 규정된 첫 원격적호스트가 낡은 문법형식을 리용한다면 그 지령은 목적지를 rhost1 로 해석하기 때문에 다음 형식으로 되여야 한다.

rcp ruser1 @ rhost1 : path1 rhost2 . ruser2 : path2

공통적으로 만나게 되는 문제는 두 원격파일들이 원격적사용자를 규정하는 원격목적지에 복사된다는것이다. 만일 두개의 원천원격체계인 rhost1 과 rhost2 를 가진다면 원격목적지에 대하여 이 매개가 다른 형식을 요구할 때 다음과 같은 지령으로 되여야 한다.

rcp rhost1 : path1 rhost2 : path2 rhost3 : ruser3 : path3

이것은 원천체계들중의 어느 하나의 체계에서 확실히 실패할것이다. 두개의 개별적지령으로 이런 전송을 수행해야 한다.

저자

rcp 는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

관련항목

cp(1), ftp(1), remsh(1), remshd(1M), rcmd(3), hosts(4), hosts . equiv(4), sis(5)
Using Internet Services 에서 ftp 장을 참고하면 된다.

remsh

remsh - 원격호스트와 접속을 하고 지령을 수행시킨다.

---

remsh(1)

이름

remsh - 원격적인 쉘로부터 수행한다.

형식

remsh host [-l username] [-n] command
host [-l username] [-n] command

rexec host [-l username] [-n] command

해설

remsh 는 규정된 호스트에 접속하고 규정된 지령을 수행시킨다. 호스트이름은 gethostbyname()로 리해되는 공식적이름 혹은 별명으로 될수 있다. remsh 는 표준입력(stdin)을 원격지령에 복사하고 원격지령의 표준출력을 그것의 표준출력(stdout)에 복사하며 원격지령의 표준오유를 그것의 표준오유(stdrr)에 복사한다. 지체, 중단, 포기, 종결, 잘리운 파이프신호들은 원격지령에 전달된다. remsh 는 원격지령의 stdout 와 stdrr 와 관련된 소케트가 닫길 때 끝나게 된다. 이것은 원격지령이 수행될 때 remsh 가 표준적으로 종결된다는것을 의미한다.

기정으로 remsh 는 규정된 지령을 수행시킬 때 다음과 같은 경로를 리용한다.

/usr/bin:/usr/ccs/bin:/usr/bin/x11:

remsh 는 원격지령을 수행하기 위하여 선택항목 -c 로 기정적인 원격가입의 쉘을 리용한다. 만일 기정의 원격쉘이 csh 이면 csh 는 이 지령이 수행되기전에 원격파일 .cshrc 을 원천파일로 한다. remsh 는 말단대면부를 요구하는 지령 (vi 와 같은) 혹은 그것의 표준오유를 읽는 지령들(more 와 같은)의 수행에 리용될수 없다. 이 경우 remsh 대신에 rlogin 과 telnet 를 리용하여야 한다.

리용된 원격계산서이름은 선택항목 -l 로 다중원격이름을 규정하지 않는한 국부계산서이름과 같다. 이 원격계산서이름은 원래 계산서와 같아야 한다. 지령과 통과암호규정에 대한 그 어떤 봉사도 없다. 동등한 호스트들과 그 규정방법에 대한 더 상세한 내용을 알려면 hosts.equiv(4)를 참고하여야 한다. 원격호스트에서 remshd 가 리용한 파일들은 /etc/hosts.equiv 와 $HOME/.rhosts 이다.

만일 지령이 규정되지 않았다면 단일한 지령을 수행할대신에 rlogin 을 리용하여 원격호스트에 가입되게 된다. 지령행에 입력된 rlogin 의 임의의 선택항목들은 rlogin 에서 취급되게 된다. 만일 지령이 규정되면 rlogin 에 규정된 선택항목들은 remsh 에서 무시된다.

기정으로 remsh 는 그것의 표준입력을 읽고 remsh 가 원격지령의 입력을 요구하는가를 결정하는 방법이 없기때문에 원격지령에 그것을 보낸다. 선택항목 -n 는 /dev/null 로부터 remsh 에로 표준입력을 방향바꾸기 한다. 이것은 지령 remsh 을 포함하는 쉘스크립트를 수행시킬 때 유용하며 다른 경우에 remsh 는 그것에 대하여 필요 없는 입력을 리용할수 있다. 선택항목 -n 은 일감조종쉘인 /usr/bin/csh 혹은 /usr/bin/ksh 로부터 배경에서 remsh 가 수행될 때에도 편리하다. 다른 경우에 remsh 는 정지되며 원격적지령에 대한 말단건반으로부터 입력을 대기하게 된다. /usr/bin/sh 는 일감이 배경에서 수행될 때 /dev/null 로부터 그 입력을 자동적으로 방향바꾸기한다.

원격호스트들에 대한 호스트이름은 등록부 /usr/hosts 에서 지령(remsh 와 련결된)으로 될수 있다. 만일 이 등록부가 $PATH 환경변수로 규정되면 remsh 는 생략될 수 있다. 실례로 만일 원격호스트가 원격호스트의 이름이라면 /usr/host/remotehost 는 remsh 와 련결된다. 만일 /usr/hosts 가 찾기경로에 있다면 지령

remotehost command

는 원격호스트에서 command 를 수행하며 지령

remotehost

는

rlogin remotehost

와 같다.

remsh 와 련결된 지령 rexec 은 지령이 수행될 때 서고루틴 rexec()과 rexecd 를 리용하는것외에 remsh 와 같이 작업한다. rexec 는 인증을 위하여 hosts.equiv 를 리용할대신에 지령을 수행하기전에 통과암호를 예고한다. 원격계산서에 대한 통과암호가 알려 져 있으나 remsh 에 불충분한 허락이 있기때문에 통과암호는 예고되였을것이다.

실례들

인증되지 않은 쉘의 메타기호들은 국부적호스트에서 해석되고 인증된 메타기호들은 원격적호스트에서 해석된다. 이리하여 다음의 지령행

remsh otherhost cat remotefile >> localfile

은 지령행

remsh otherhost cat remotefile">>" otherremotefile

이 remotefile 을 원격파일 otherremotefile 에 추가할동안 원격파일 remotefile 를 국부파일 localfile 에 추가한다.
만일 원격쉘이 /usr/bin/sh 이면 다음의 지령행은 원격지령을 수행하기전에 그 원격지령에 환경을 설정한다.

remsh otherhost. . profile 2 > &-\;command

2 > &-는 stdin 과 stdout 가 말단이 아닐 때 .profile 수행으로 발생된 오유통보들을 버리게 한다.
다음의 지령행은 국부체계의 배경에서 remsh 를 수행하고 원격지령의 출력은 비동기적으로 말단에 오게 된다.

remsh otherhost ?n command &

배경 remsh 는 원격지령을 수행할 때 완성된다.
다음 지령행은 remsh 가 원격지령의 완성을 기다리지 않고 직접 귀환하게 한다.

remsh otherhost ?n "command 1> & -2>&- &"

만일 원격체계에서 가입쉘이 csh 이면 다음의 지령을 우의 지령대신에 리용하여야 한다.

remsh otherhost ?n "sh ?c\"command 1 >& - 2 >& - &\""

귀환값

만일 remsh 가 2 차소케트접속설정에서 실패한다면 2 를 귀환한다. 만일 다른 어떤 방법으로 실패하였다면 1 을 귀환한다. 그것은 remsh 가 접속설정을 완전히 성공하고 그리고 원격지령을 완성하였다면 0 을 귀환한다. remsh 의 귀환값은 원격지령의 귀환값과는 관계없이 전달되게 된다.

진단

아래에서 목록화된 오유가 아닌 다른 오유들은 remsh 에서 리용되는 rcmd()와 rresvport()서고함수로 발생될수도 있다. 이 오유들의 앞에는 그것을 발생시킨 서고함수이름이 놓이게 된다. remsh 는 다음과 같은 진단통보를 발생시킬수 있다.

rlogin: … 가입할 때의 오유이다. (rlogin 은 사용자가 수행해야 할 임의의 지령을 규정하지 않을 때 수행된다.) 그다음에 수행이 실패한 원인을 규정한 오유통보가 놓인다.

shell/tcp:unknown service
봉사기규정 "shell"은 파일 /etc/services에 제출되지 않는다.

can't establish staderr
remsh는 stderr를 위한 2차소케트접속을 설정할수 없다.

<system coll>:…
오유는 체계호출을 수행할 때 발생된다. 이 오유에 추가된 통보는 실패의 원인을 규정한다.

There is no entry for you(user ID uid)in/etc/paawd
만일 통과암호파일에서 등록부가 실수로 인하여 제거되였다면 체계관리자에 의하여 검사되여야 한다.

경고

안전성리유로 파일 /etc/hosts.equiv 와 .rhosts 들은 빌 때에도 존재하여야 하며 파일소유자에 의해서만 읽을수 있고 쓸수 있게 되여야 한다. 또한 필요할 때 임의의 통과암호를 포함한 모든 통보는 두 호스트들사이에 전달된다는것을 알아야 한다.
만일 remsh가 호상작용지령으로 수행된다면 이 지령은 벗어 나지 못한다.

종속성

remsh 는 BSD 체계에서 rsh 와 같은 봉사프로그람이다. 이미 존재하는 system V 의 지령 rsh(제한된 shell)와 이름이 일치되므로 이름은 변경되였다.

저자

remsh는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

파일들

/usr/hosts/* 호스트이름으로만 기동된 지령의 방안에 대해서만 리용한다.

관련항목

rlogin(1), renshd(1M), rexecd(1M), gethostent(3N), rcmd(3N), rexec(3N), hosts.equiv(4), hosts(4)

remsh(1) – Kerberos 인증으로 인터네트봉사를 안전하게 보장한다. remsh(1)

이름

remsh – 원격쉘로부터 수행한다.

형식

remsh host [-l username] [-f/F] [-R realm] [-p] [-n] command
host [-l username] [-f/F] [-k realm] [-p] [-n] command
rexec host [-l username] [-n] command

해설

remsh 는 규정된 호스트에 접속하고 규정된 지령을 수행시킨다. 호스트이름은 gethostbyname( )로 리해되는 공식적이름 혹은 별명으로 될수 있다. remsh 는 표준입력(stdin)을 원격지령에 복사하고 원격지령의 표준출력을 그것의 표준출력(stdout)에 복사하며 원격지령의 표준오유를 그것의 표준오유(stdrr)에 복사한다. 지체, 중단, 포기, 종결, 잘리운 파이프신호들은 원격지령에 전달된다. remsh 는 원격지령의 stdout, stdrr 와 관련된 소케트가 닫길 때 끝나게 된다. 이것은 원격지령이 수행될 때 remsh 가 표준적으로 종결된다는것을 의미한다.

기정으로 remsh 는 규정된 지령을 수행시킬 때 다음과 같은 경로를 리용한다.

/usr/bin:/usr/ccs/bin:/usr/bin/x11:

remsh 는 원격지령을 수행하기 위하여 선택항목 ?c 로 기정적인 원격가입의 쉘을 리용한다. 만일 기정의 원격쉘이 csh 이면 csh 는 이 지령이 수행되기전에 원격파일 .cshrc 을 원천파일로 한다. remsh 는 말단대면부를 요구하는 지령 (vi 와 같은) 혹은 그것의 표준오유를 읽는 지령들(more 와 같은)의 수행에 리용될수 없다. 이 경우에는 remsh 대신에 rlogin 과 telnet 를 리용하여야 한다.

리용된 원격계산서의 이름은 선택항목 -l 로 다중원격이름을 규정하지 않는한 국부계산서의 이름과 같다.
추가적으로 원격호스트의 계산서이름은 원격호스트가 Kerberos V5 망의 인증에서 입장할 때 안전환경이 있거나 없는가에 따라서 달라 지는 규칙과 일치되여야 한다. 비안전환경 혹은 전통적인 환경에서 원격계산서의 이름은 원래 계산서의 이름과 같아야 한다. 지령과 통과암호의 규정에 대해서는 준비되여 있지 않다. 동등한 호스트들과 그것을 규정하는 방법에 대한 자세한 내용을 알려면 hosts.equiv(4)를 참고하여야 한다. 원격호스트에서 remsh 로 검사되는 파일들은 /etc/hosts.equiv 와 $HOME/.rhosts 이다.

Kerberos V5 망의 접근환경에서 국부적호스트는 원격계산서의 이름이 인정되기 위해서는 검사되기전에 성과적으로 인증되여야 한다. 인증수단은 원격호스트에서 remshd 의 기동에 리용될 지령행인수들(-K, -R, -r, -k)에 의존한다. Kerberos 인증과 인정에 대한 자세한 통보를 알려면 인터네트봉사의 안전에 대한 기본페지에서 sis(5)와 remshd(1M)를 참고하면 된다.

안전환경 혹은 Kerberos V5 에 기초한 환경에서 다음과 같은 지령행의 선택항목들을 리용할수 있다.

-f　　원격체계에 입장권을 담보하는 입장권(TGT)을 전송한다. TGT 는 여기로부터 전송될수 없다.

-F　　원격소케트에 TGT 를 전송하고 이로부터 다른 원격체계에 전송할수 있다. -f 와 -F 는 서로 배타적이다.

-k realm　구성파일 krb.realms 에서 규정된것과 같은 원격호스트의 기정지대대신에 규정된 realm 으로 원격호스트로부터 입장권을 얻는다.

-p　　Kerberos 의 인정을 하지 못하게 한다.

만일 지령이 규정되지 않는다면 단일지령을 수행할대신에 rlogin 을 리용하여 원격호스트에 가입하게 될것이다. 지령행에서 입력된 rlogin 의 임의의 선택항목들은 rlogin 에 전달된다. 만일 지령이 없고 선택항목 -p 가 규정되면 rlogin 은 Kerberos 인정(혹은 안전접근)이 요구되지 않는다는것을 지적하기 위하여 -p 를 가지고 수행되여야 한다. 만일 통과암호가 요구된다는것을 의미한다면 통과암호는 명백히 보내 질것이다. 만일 지령이 규정되면 rlogin 에서 선택항목들의 규정은 remsh 에 의하여 무시된다.

만일 지령과 선택항목 -n 이 규정되면 표준입력은 /dev/null 에 의하여 remsh 에서 방향바꾸기를 한다. 만일 -n 이 규정되지 않으면(기정인 경우) remsh 는 그것의 표준입력을 읽고 원격지령에 입력을 보낸다. 왜냐하면 remsh 가 원격지령의 입력을 요구하는가를 결정할 방법이 없기때문이다. 이 선택항목은 지령 remsh 를 포함하는 쉘스크립트를 수행시키는데 유용하다. 다른 경우에 remsh 는 그에 대하여 불필요한 입력을 리용한다. 선택항목 -n 은 일감조종쉘인 /usr/bin/csh 혹은 /usr/bin/ksh 로부터 배경에서 remsh 가 수행될 때에도 편리하다. 다른 경우에 remsh 는 정지되며 원격적지령에 대한 말단건반으로부터 입력을 대기하게 된다. /usr/bin/sh 는 일감이 배경에서 수행될 때 /dev/null 로부터 그 입력을 자동적으로 방향바꾸기를 한다.
원격호스트들에 대한 호스트이름은 등록부 /usr/hosts 에서 지령(remsh 와 련결된)으로 될수 있다. 만일 이 등록부가 $PATH 환경변수로 규정되면 remsh 는 생략될수 있다. 실례로 만일 원격호스트가 원격호스트의 이름이라면 /usr/host/remotehost 는 remsh 와 련결된다. 만일 /usr/hosts 가 찾기경로에 있다면 지령

```
remotehost command
```

는 원격호스트에서 command 를 수행하며 지령

```
remotehost
```

는

rlogin remotehost

와 같다.

remsh와 련결된 지령 rexec은 지령이 수행될 때 서고루틴 rexec( )과 rexecd를 리용하는것외에 remsh와 같이 작업한다. rexec는 인증을 위하여 hosts.equiv를 리용할대신에 지령을 수행하기전에 통과암호를 예고한다. 원격계산서에 대한 통과암호가 알려 져 있으나 remsh에 불충분한 허락이 있기때문에 통과암호는 예고되였을것이다.

실례들

인증되지 않은 쉘의 메타기호들은 국부적호스트에서 해석되고 인증된 메타기호들은 원격적호스트에서 해석된다. 이리하여 다음의 지령행

remsh otherhost cat remotefile >> localfile

은 지령행

remsh otherhost cat remotefile">>" otherremotefile

이 remotefile을 원격파일 otherremotefile에 추가할동안 원격파일 remotefile를 국부파일 localfile에 추가한다.

만일 원격쉘이 /usr/bin/sh이면 다음의 지령행은 원격지령을 수행하기전에 그 원격지령에 환경을 설정한다.

remsh otherhost . . profile 2 > &-\;command

2 > &-는 stdin과 stdout가 말단이 아닐 때 .profile수행으로 발생된 오유통보들을 버리게 한다.

다음의 지령행은 국부체계의 배경에서 remsh를 수행하고 원격지령의 출력은 비동기적으로 말단에 오게 된다.

remsh otherhost ?n command &

배경 remsh는 원격지령을 수행할 때 완성된다.

다음 지령행은 remsh가 원격지령의 완성을 기다리지 않고 직접 귀환하게 한다.

remsh otherhost ?n "command 1> & -2>&-&"

만일 원격체계에서 가입쉘이 csh이면 다음의 지령을 우의 지령대신에 리용하여야 한다.

remsh otherhost ?n "sh ?c\"command 1 >& - 2 >& - &\""

귀환값

만일 remsh 가 2 차소케트접속설정에서 실패한다면 2 를 귀환한다. 만일 다른 어떤 방법으로 실패하였다면 1 을 귀환한다. 그것은 remsh 가 접속설정을 완전히 성공하고 그리고 원격지령을 완성하였다면 0 을 귀환한다. remsh 의 귀환값은 원격지령의 귀환값과는 관계없이 전달되게 된다.

진단

아래에서 목록화된 오유를 벗어 나는 오유들은 remsh 에서 리용되는 rcmd()와 rresvport()서고함수에 의하여 발생될수도 있다. 이 오유들의 앞에는 그것을 발생시킨 서고함수이름이 놓이게 된다. remsh 는 다음과 같은 진단통보를 발생시킬수 있다.

rlogin: …

가입을 수행할 때 오유이다(rlogin 은 사용자가 수행해야 할 임의의 지령을 규정하지 않을 때 수행된다.). 그다음에 수행이 실패된 원인을 규정한 오유통보가 놓인다.

shell/tcp:unknown services

쉘봉사기규정은 파일 /etc/services 에 제시되지 않는다.

can't establish staderr

remsh 는 stderr 를 위한 2 차소케트접속을 설정할수 없다.

<system call>:…

오유는 체계를 호출할 때 발생된다. 이 오유에 추가된 통보는 실패의 원인을 규정한다.

There is no entry for you(user ID uid)in/etc/paawd

만일 통과암호파일에서 실수로 인하여 등록부가 제거되였다면 체계관리자에 의하여 검사되여야 한다.

rcmd : connect : <hostname>: Connection refused

이 일반오유통보를 표시하는 리유는 원격체계의 /etc/inetd.conf 에서 쉘에 대한 등록항목이 없기때문이다. 이 등록항목은 비안전접근을 막기 위하여 제거되거나 설명문화될수 있다.

Kerberos - 규정오유들은 sis(5)에서 목록화된다.

경고

안전성리유로 파일 /etc/hosts.equiv 와 .rhosts 들은 파일내용이 없을 때에도 존재하여야 하며 파일소유자에 의해서만 읽을수 있고 쓸수 있게 되여야 한다. 또한 필요할 때 임의의 통과암호를 포함한 모든 통보는 두 호스트들사이에 전달된다는 것을 알아야 한다.

만일 remsh 가 호상작용지령으로 수행된다면 이 지령은 벗어 나지 못한다.

종속성

remsh 는 BSD 체계에서 rsh 와 같은 봉사프로그람이다. 이미 존재하는 System V 의 지령 rsh(제한된 shell)와 이름이 일치되므로 이름은 변경되였다.

저자

remsh 는 캘리포니아종합대학, 버클리에 의해 개발되였다.

파일들

/usr/hosts/* 호스트이름으로만 기동된 지령의 방안에 대해서만 리용한다.

관련항목

rlogin(1), renshd(1M), rexecd(1M), gethostent(3N), rcmd(3N), rexec(3N), hosts.equiv(4), hosts(4), sis(5)

rlogin

rlogin – 원격호스트에 가입한다.

이름

rlogin – 원격가입

형식

rlogin rhost [-7] [-88] [-ee] [-l username]

rhost [-7] [-8] [-ee] [-l username]

해설

지령 rlogin 은 국부호스트의 말단을 원격호스트(rhost)와 접속시킨다. rlogin 은 원격체계에서 가상말단으로 작용한다. 호스트이름 rhost 는 파일 /etc/hosts 에서 목록

화된 공식적이름 혹은 별명으로 될수 있다.

지령 remsh 과 류사한 방법으로 rlogin 은 사용자에게 동등한 원격호스트인 rhost 에 표준가입/통과암호렬의 전달에 의하여 가입하게 한다. 동등한 호스트와 파일 /etc/hosts.equiv 와 rhost 에서 그것을 규정하는 방법에 대한 더 많은 정보를 알려면 hosts.equiv(4)를 참고하여야 한다. 파일 /etc/hosts.equiv 와 .rhosts 의 찾기는 원격호스트에서 진행하며 파일 .rhosts 는 원격사용자의 계산서와 원격상급사용자에 의하여 소유되여야 한다.

만일 사용자의 계산서가 원격사용자의 계산서와 동일하지 않으면 원래 사용자는 원격계산서의 통과암호에 대한 예고를 한다. 이것이 실패한다면 가입이름과 통과암호는 가입에 리용된것처럼 예고한다.

현재환경변수 TERM 에서 규정된 말단형은 망을 통하여 전파되며 원격호스트에서 환경변수 TERM 의 시동값을 수정하는데 리용된다. 이것은 말단의 전송속도로 원격호스트에 전파되는데 일부 체계에서 rlogin 으로 리용된 허위말단설정을 하는데 필요하다.

모든 반사는 원격가입이 지체하지 않고 투파되도록 하기 위하여 원격지령에서 수행된다.
만일 임의의 시간에 rlogin 이 원격호스트에서 소케트접속에 쓰기 혹은 읽기에 리용되지 않는다면 접속이 닫길 때 통보는 표준오유로 출력되고 rlogin 은 끝난다.

선택항목들

rlogin 은 다음과 같은 선택항목들을 가진다. 선택항목들은 rhost 인수의 뒤에 놓인다.

-7 기호크기를 7 개 비트로 설정한다. 보낸 매 바이트의 8 번째 비트는 0 으로 설정된다(짝홀성).

-8 비트자료방식을 리용한다. 이것은 HP-UX 의 기정행동이다. 8 개 비트를 가진 기호를 리용할 때 말단은 짝홀성이 없는 경우 8 개 비트 혹은 짝홀성이 있는 경우 7 개 비트를 발생하도록 모형화되여야 한다. HP-UX 의 rlogin 의 실현은 홀수성, 짝수성으로 7 개 비트를 가진 기호를 해석하거나 짝홀성으로 8 개 비트를 가진 비 USAASCII 기호를 해석한다. 원격호스트를 적당히 재모형화하는것이 요구될수 있다. 일부 원격호스트들은 8 개 비트기호들에 대한 필요한 봉사를 제공하지 않는다. 이 경우에 만일 국부말단에서 짝홀성발생을 하지 못하게 하는것이 불가능하다면 선택항목 -7 을 리용한다.

-ee　　탈퇴기호를 e 로 설정한다. 이것은 선택항목인 문자와 인수기호를 분리하는 공백이 아니다. 탈퇴기호로 행을 시작하기 위해서는 두개의 탈퇴기호를 입력하여야 한다. 기정의 탈퇴기호는 물결표(～)이다. 일부 기호들은 ^S, ^Q, BS 와 같은 말단모형과 모순될수 있다. 탈퇴기호로서 이 기호들중의 어느한 기호를 리용할수 없으므로 원격호스트와 통신에서는 문제가 제기될수 있다.

-l username　　원격호스트에서 사용자가입이름을 username 으로 설정한다. 기정이름은 rlogin 을 기동시킨 사용자의 현재계산서의 이름이다.

탈퇴렬

rlogin 은 형식 ex 로 된 두개 기호의 탈퇴렬을 리용한다. 여기서 e 는 탈퇴기호이고 x 는 아래에서 보여 주는 코드기호이다. 탈퇴렬은 입력행의 시작에서만 식별된다. 기정탈퇴기호는 물결표(～)이다. 이것은 선택항목 -e 로 변경시킬수도 있다.

탈퇴기호렬은 다음과 같다.

ey　　만일 y 가 아래에서 서술되는 한개 코드기호의 NOT 이면 원격호스트에로 전달되고 기호로서 탈퇴기호와 y를 전달한다.

ee　　원격호스트에 한개 기호로서 탈퇴기호렬을 전달한다.

e .　　원격호스트로부터 접속을 차단한다.

e!　　국부호스트에서의 부분쉘에로 탈퇴한다. 원격호스트에로 귀환을 위하여 나가는데 리용된다.

만일 rlogin 이 일감조종에 봉사하는 쉘로부터 수행된다면 탈퇴렬은 rlogin 의 지연에 리용될수 있다. 다음의 탈퇴렬은 ^Z 와 ^Y 가 각각 사용자의 기호 susp 와 dsusp 들로 설정된다고 간주한다.

e^Z　　rlogin 의 가입기간을 연기하고 rlogin 은 기동된 쉘에로 사용자를 귀환시킨다. rlogin 일감은 지령 fg 로 규정될수 있다. e^Z 는 rlogin 의 두 프로쎄스 즉 원격가입에 사용자입력의 전송과 원격가입으로부터 출력표시프로쎄스를 지연시킨다.

e^Y　　rlogin 의 가입기간을 지연시키고 rlogin 은 기동된 쉘로 사용자를 귀환시킨다. rlogin 일감은 지령 fg 로 규정될수 있다. e^Y 는 입력파일만 지연시키고 원격가입으로부터 출력을 계속 표시하게 한다.

만일 유일한 탈퇴기호를 설정하지 않고 원격가입을 한다면 선택된 목적지에 도달할 때까지 탈퇴기호를 반복할수 있다(실례로 호스트 A 로부터 호스트 B 에

가입하고 다음에 호스트 B 로부터 호스트 C 에 가입한다.). 실례로 첫번째 탈퇴기호 e 는 호스트 A 에 탈퇴기호로 보내고 두번째 탈퇴기호 e 는 호스트 A 에 의하여 표준기호로 전달되며 호스트 B 에 탈퇴기호로 보낸다. 세번째 e 는 호스트 A 와 B 에 의하여 표준기호로 전달되고 호스트 C 에 의하여 표준기호로 접수된다.

지령으로서 원격호스트이름

체계관리자는 한개 지령으로서 원격호스트이름(rhost)의 리용을 허용하면서 /usr/hosts/rhost 에 remsh 를 련결하여 원격호스트(rhost)에 더 편리한 접근을 보장해 준다. 실례로 만일 remotehost 가 원격호스트의 이름이고 /usr/hosts/remotehost 가 remsh 와 련결되며 /usr/hosts 가 자기의 찾기 경로에 있다면 지령

```
remotehost
```

는

```
rlogin remotehost
```

와 같다.

귀환값

rlogin 은 표준오유에로 오유통보를 보내고 만일 오유가 원격호스트와 접속이 완성되기전에 나타났다면 령아닌 값을 귀환한다. 다른 경우에는 0 을 귀환한다.

진단

진단은 국부호스트 및 원격호스트들로부터 나타날수 있다. 접속이 완전히 설정되기전에 국부호스트에 나타나는 오유들은 표준오유에 출력된다. 일단 접속이 설정되면 원격호스트로부터 임의의 오유통보는 임의의 다른 자료와 같이 표준출력에 출력된다.

login/tcp:unknown service

rlogin 은 /etc/service 자료기지의 파일에 목록화된 가입봉사를 찾기 위하여 리용될수 없다.

there is no entry for you(user ID username) in /etc/passwd

rlogin 은 통과암호파일에서 사용자 ID 를 찾을수 없다.
다음 걸음은 체계관리자가 조절하여야 한다.

system call: . . .

rlogin 이 체계를 호출하였을 때 나타나는 오유이다. 오유통보에 대하여 더 잘 알려면 편람에서 대응하는 항목들을 참고하면 된다.

실례들

원격호스트 remote에 같은 사용자로서 가입하려면 지령

rlogin remote

을 주면 된다.

탈퇴기호를 !로 설정하고 7개 비트자료의 접속을 리용하며 호스트 remhost에서 사용자대리인으로 가입하려고 한다면 지령

rlogin remhost -e! -7 -l quest

를 준다.

체계관리자가 /usr/hosts 에 련결설정을 가진다면 다음의 지령은 앞의 지령과 같다.

remhost -e! -7 -1 quest

경고

안전성을 목적으로 파일 /etc/hosts.equiv 와 .rhosts 들은 파일의 내용들이 없을 때에도 존재해야 한다. 이 파일들은 파일소유자에 의해서만 읽을수 있고 쓸수 있어야 한다. 더 많은 통보를 알려면 host.equiv(4)를 참고하여야 한다.

질문되는 임의의 통과암호를 포함한 모든 정보는 두 호스트들사이에 보호없이 전달된다.

rlogin 은 사용자가 국부체계에 stty brkint 을 설정하였는가, 설정하지 않았는가에 무관계하게 원격체계에 중단신호로서 Break 건을 전송할수 없다. 대신에 지령 stty intr c로 SIGINT에 부과된 건을 리용할수 있다.

저자

rlogin은 캘리포니아종합대학, 버클리에 의하여 개발되였다.

파일들

$HOME/.rhosts	사용자개별의 동등한 목록
/etc/hosts.equiv	동등한 호스트들의 목록
/usr/hosts/*	지령의 rhost 방안에 대하여 리용된다.

관련항목

csh(1), ksh(1), login(1), remsh(1), sh(1), sh_bourne(1), sh_posix(1), stty(1), telnet(1), rlogind(1M), hosts(4), hosts.equiv(4), inctd.conf(4), services(4), termio(7), tty(4), sis(5)

rlogin(1) Kerberos 인증 rlogin(1)과의 인터네트봉사를 안전하게 한다. rlogin(1)

이름

rlogin – 원격가입

형식

rlogin rhost [-7] [-8] [-ee] [-f/F] [-k realm] [-l username] [-p]
rhost [-7] [-8] [-ee] [-f/F] [-k realm] [-l username] [-p]

해설

지령 rlogin 은 국부호스트의 말단을 원격호스트(rhost)와 접속시킨다. rlogin 은 원격체계에서 가상말단으로 작용한다. 호스트이름 rhost 는 파일 /etc/hosts 에서 목록화된 공식적이름 혹은 별명으로 될수 있다.

현재환경변수 TERM 에서 규정된 말단형은 망을 통하여 보급되며 원격호스트에서 환경변수 TERM 의 시동값을 수정하는데 리용된다. 이것은 말단의 전송속도로 원격호스트에 전파되며 일부 체계에서 rlogin 으로 리용된 허위말단설정에 요구된다.

모든 **반사**(Echoing)는 원격가입이 지체하지 않고 투과(transparent)되도록 하기 위하여 원격지령에서 수행된다.

만일 임의의 시간에 rlogin 이 원격호스트에서 소케트접속에 쓰거나 읽기에 리용되지 않는다면 접속이 닫길 때 통보는 표준오유로 출력되고 rlogin 은 끝난다.

Kerberos V5 망체계의 인증환경에서 rlogin 은 원격호스트에 접속을 인증받기 위하여 Kerberos V5 규약을 리용한다. 만일 인증이 성공하면 사용자인정은 rlogind 로 선택한 선택된 지령행인수들(즉 -K, -R, -k, -r)에 따라서 수행될것이다. 통과암호예고를 하지 않고 또한 망이 통과암호를 리용하지 않도록 하였으므로 사용자에게 통과암호는 요구되지 않는다. Kerberos 인정과 인증에 대하여 더 많은 통보를 알려면 인터네트봉사안전에 대한 기본페지의 sis(5), rlogind(1M)을 참고하여야 한다.
비록 Kerberos 인정과 인증은 적용될수 있지만 Kerberos 수단은 가입기간에 적용될수 없다. 자기의 호스트와 원격호스트사이에 전송된 모든 정보는 망체계에서 명백히 보내여 진다.

선택항목들

rlogin 은 다음과 같은 선택항목들을 가진다. 선택항목들은 rhost 인수의 뒤에 놓인다.

-7 기호크기를 7 개 비트로 설정한다. 보낸 매 바이트의 8 번째 비트는 0 으로 설정된다(짝홀성공간).

-8 비트자료방식을 리용한다. 이것은 HP-UX 의 기정행동이다. 8 개 비트를 가진 기호를 리용할 때 말단은 짝홀성이 없는 경우에는 8 개 비트 혹은 짝홀성이 있는 경우 7 개 비트를 발생하도록 모형화되여야 한다. HP-UX 의 rlogin 의 실현은 홀수성,

짝수성으로 7개 비트를 가진 기호를 해석하거나 짝홀성으로 8개 비트를 가진 비 USAASCII 기호를 해석한다. 원격호스트를 적당히 재모형화하는것은 필요될수 있다. 일부 원격호스트들은 8개 비트기호들에 대한 필요한 봉사를 제공하지 않는다. 이 경우에 만일 국부말단에서 짝홀성발생을 하지 못하게 하는 것이 불가능하다면 선택항목 -7을 리용한다.

-ee 탈퇴기호를 e로 설정한다. 이것은 선택항목인 문자와 인수기호를 분리하는 공백이 아니다. 탈퇴기호로 행을 시작하기 위하여서는 두개의 탈퇴기호를 입력하여야 한다. 기정의 탈퇴기호는 물결표(～)이다. 일부 기호들은 ^S, ^Q, BS와 같은 말단모형과 모순될수 있다. 탈퇴기호로서 이 기호들중의 어느한 기호를 리용할수 없으므로 원격호스트와 통신에서 문제가 제기될수 있다(stty(1)과 tty(7)을 참고).

-f 원격체계에서 입장표(ticket)를 담보하며 그 입장표를 전송한다. TGT는 여기로부터 전송할수 없다.

-F 원격체계에 TGT를 전송하지만 여기로부터 다른 원격체계에로는 전송할수 없다. -f와 -F들중 어느 하나를 리용한다.

-k realm 형파일 krb.realms에서 규정된것과 같이 원격호스트의 기정지대대신에 규정된 realm으로 원격호스트로부터 입장표를 얻을수 있다.

-l username 원격호스트에서 사용자의 가입이름을 username으로 설정한다. 기정이름은 rlogin을 기동시킨 사용자의 현재계산서의 이름이다.

-P Kerberos 인증을 하지 못하게 한다. Kerberos V5에 기초한 안전환경에서만 적용될수 있다. 이 선택항목이 규정될 때에는 통과암호가 요구되며 명백히 망체계을 통하여 보내여 진다. 표준렬 login/password을 전달하려면 remsh와 같은 지령으로 동등한 계산서를 리용하여 원격호스트에 가입하여야 한다. 이에 대한 자세한 통보를 알려면 hosts.equiv(4)를 참고하면 된다.

탈퇴렬

rlogin은 형식 ex로 된 두개 기호의 탈퇴렬을 리용한다. 여기서 e는 탈퇴기호이고 x는 아래에서 보여 주는 코드기호이다. 탈퇴렬은 입력행의 시작에서만 식별된다. 기정탈퇴기호는 물결표(～)이다. 이것은 선택항목 -e로 변경시킬수 있다.

탈퇴기호렬은 다음과 같다.

ey 만일 y 가 아래에서 서술되는 한개 코드기호의 NOT 이면 원격호스트에로 전달되고 기호로서 탈퇴기호와 y를 전달한다.

ee 원격호스트에 한개 기호로서 탈퇴기호렬을 전달한다.

e . 원격호스트로부터 접속을 차단한다.

e! 국부호스트에서의 부분쉘에로 탈퇴한다. 원격호스트에로 귀환을 위하여 나가는데 리용된다.

만일 rlogin 이 일감조종에 봉사하는 쉘로부터 수행된다면 탈퇴렬은 rlogin 의 지연에 리용될수 있다. 다음의 탈퇴렬은 ^Z 와 ^Y 가 각각 사용자의 기호 susp 와 dsusp 들로 설정된다고 간주한다.

e^Z rlogin 의 가입기간을 연기하고 rlogin 은 기동된 쉘에로 사용자를 귀환시킨다. rlogin 일감은 지령 fg 로 규정될수 있다. e^Z 는 rlogin 의 두 프로쎄스 즉 원격가입에 사용자입력의 전송과 원격가입으로부터 출력표시프로쎄스를 지연시킨다.

e^Y rlogin 의 가입기간을 지연시키고 rlogin 은 기동된 쉘로 사용자를 귀환시킨다. rlogin 일감은 지령 fg 로 규정될수 있다. e^Y 는 입력파일만 지연시키고 원격가입으로부터 출력을 계속 표시하게 한다.

만일 유일한 탈퇴기호를 설정하지 않고 원격가입을 한다면 선택된 목적지에 도달할 때까지 탈퇴기호를 반복할수 있다. (실례로 호스트 A 로부터 호스트 B 에 가입하고 다음에 호스트 B 로부터 호스트 C 에 가입한다.) 실례로 첫번째 탈퇴기호 e 는 호스트 A 에 탈퇴기호로 보내고 두번째 탈퇴기호 e 는 호스트 A 에 의하여 표준기호로 전달되며 호스트 B 에 탈퇴기호로 보낸다. 세번째 e 는 호스트 A 와 B 에 의하여 표준기호로 전달되고 호스트 C 에 의하여 표준기호로 접수된다.

지령으로서 원격호스트이름

체계관리자는 한개 지령으로서 원격호스트이름(rhost)의 리용을 허용하면서 /usr/hosts/rhost 에 remsh 를 련결하여 원격호스트(rhost)에 더 편리한 접근을 보장해 준다. 실례로 만일 remotehost 가 원격호스트의 이름이고 /usr/hosts/remotehost 가 remsh 와 련결되며 /usr/hosts 가 자기의 찾기 경로에 있다면 지령

remotehost

는

rlogin remotehost

와 같다.

귀환값

rlogin 은 표준오유에로 오유통보를 보내고 만일 오유가 원격호스트와 접속이 완성되기전에 나타났다면 령이아닌 값을 귀환한다. 다른 경우에는 0 을 귀환한다.

진단

진단은 국부 및 원격호스트들로부터 나타날수 있다. 접속이 완전히 설정되기전에 국부호스트에 나타나는 오유들은 표준오유에 출력된다. 일단 접속이 설정되면 원격호스트로부터 임의의 오유통보는 임의의 다른 자료와 같이 표준출력에 출력된다.

login/tcp:unknown service

rlogin 은 /etc/service 자료기지의 파일에 목록화된 가입봉사를 찾기 위하여 리용될수 없다.

there is no entry for you(user ID username) in /etc/passwd

rlogin 은 통과암호파일에서 사용자 ID 를 찾을수 없다.
다음 걸음은 체계관리자가 조절하여야 한다.

system call**: . . .**

rlogin 이 체계를 호출하였을 때 나타나는 오유이다. 오유통보에 대하여 더 잘 알려면 편람에서 대응하는 항목들을 참고하면 된다.

rcmd : connect <hostname> : Connection reguied

원격체계에서 /etc/inetd.sonf 에 가입을 위한 등록항목이 없기때문에 일반 오유통보가 표시된다. 이 등록항목은 비안전접근을 막기 위하여 제거되거나 설명문화될수 있다.

Kerberos – 특정오유들은 sis(5)에서 목록화된다.

실례들

원격호스트 remote 에 같은 사용자로서 가입하려면 지령을 다음과 같이 준다.

rlogin remote

탈퇴기호를 !로 설정하고 7 개 비트자료의 접속을 리용하며 호스트 remhost 에서 사용자대리인으로 가입하려고 한다면 지령을 다음과 같이 준다.

rlogin remhost -e! -7 -l quest

체계관리자가 /usr/hosts 에 련결설정을 가진다면 다음의 지령은 앞의 지령과 같다.

remhost -e! -7 -1 quest

경고

안전성을 목적으로 파일 /etc/hosts.equiv 와 .hosts 들은 파일의 내용들이 없을 때에도 존재해야 한다. 이 파일들은 파일소유자에 의하여서만 읽을수 있고 쓸수 있어야 한다. 더 많은 통보를 알려면 host.equiv(4)를 참고하여야 한다.

질문되는 임의의 통과암호를 포함한 모든 정보는 두 호스트들사이에 보호없이 전달된다. Kerberos V5 망체계의 인정환경에서 통과암호로 보호되도록 망체계를 통하여 전송되지 않는다.

rlogin 은 사용자가 국부체계에 stty brkint 을 설정하였는가, 설정하지 않았는가에 무관계하게 원격체계에 중단신호로서 Break 건을 전송할수 없다. 대신에 지령 stty intr c 로 SIGINT 에 부과된 건을 리용할수 있다.

저자

rlogin 은 캘리포니아종합대학, 버클리에 의하여 개발되였다.

파일들

$HOME/.rhosts	사용자개별의 동등한 목록
/etc/hosts.equiv	동등한 호스트들의 목록
/usr/hosts/*	지령의 rhost 방안에 대하여 리용된다.

관련항목

csh(1), ksh(1), login(1), remsh(1), sh(1), sh_bourne(1), sh_posix(1), stty(1), telnet(1), rlogind(1M), hosts(4), hosts.equiv(4), inctd.conf(4), services(4), termio(7), tty(4), sis(5)

route

route – 망의 **경로화표**(Routing Table)들에 대한 조작을 한다.

---

route(1M)

이름

route – 수동적으로 경로화표들을 조작한다.

형식

/usr/sbin/route [-f] [-n] [-p pmth] ass [net | host] destination
[network mask] getway [count]

/usr/sbin/route [-f] [-n] delete [net | host] destination
[network mask] getway [count]

/usr/sbin/route -f [-n]

해설

지령 route 은 수동적으로 망경로화표들을 조작한다. 사용자는 적당한 특정권을 가져야 한다.

부분지령들

다음과 같은 부분지령들을 리용할수 있다.

add 망경로화표에 규정된 호스트 혹은 망경로를 추가한다. 만일 경로가 이미 존재하면 통보는 인쇄되며 아무런 변경도 진행되지 않는다.

delete 망경로화표로부터 규정된 호스트 혹은 망경로를 제거한다.

선택항목들

route 는 다음과 같은 선택항목들과 인수들을 식별한다.

-f 관문의 원격호스트를 규정하는 모든 경로표의 등록항목들을 제거한다. 만일 이 인수가 부분지령들중의 어느한 지령과 함께 리용되면 등록항목은 부분지령이 처리되기전에 제거된다.

-n 기정으로 인쇄되는 기정망주소를 제외하고 인터네트점표시에서 임의의 호스트와 망주소들을 인쇄한다.

-p pmtu 정적인 호스트경로에서 경로의 최대전송단위(MTU)의 값을 규정한다. 허용되는 최소값은 68byte 이고 최대값은 이 경로로 나가는 대면부의 MTU 이다. 이 선택항목은 오직 호스트경로의 추가에서만 적용된다. 다른 모든 경우에 이 선택항목은 무시되며 체계에서 작용하지 않는다. pmtu 를 0 으로 주면 호스트경로의 MTU 를 발견하지 못하게도 할수 있다.

net 혹은 host 목적지주소의 형이다. 만일 이 인수가 생략되면 특이한 호스트의 경로는 목적지와 관련된 인터네트주소의 해석으로 망에서 구별된다. 만일 목적지가 INADDR_ANY(0)의 국부주소부분을 가지면 경로를 망이라고 간주한다. 다른 경우에 그것은 호스트의 경로로 취급된다.

destination 파케트들이 경로화될 목적지호스트체계이다. 목적지는 다음과

같은것들중의 하나로 될수 있다.

- 호스트이름(공식이름 혹은 별명. gethost byname(3N)을 참고)
- 망이름(공식이름 혹은 별명. gethostby name (3N)을 참고)
- 점표시에서 인터네트주소(inet(3N)을 참고)
- 그룹기호의 관문경로로 의미를 가지는 기정실마리어(routing(9)를 참고)

netmask

mask 파케트가 경로화될 망주소의 계산을 위하여 목적지주소와 자리별론리적될 마스크이다. 마스크는 점표시인터네트주소로 ox가 앞에 놓이는 한개의 16진수로 규정되거나 망표에서 목록화된 허위망이름으로 규정될수 있다(network(4)를 참고). 마스크의 길이는 32bit 마당의 가장 왼쪽 비트의 위치로부터 시작하여 련속적으로 놓인 1들의 개수이고 목적지주소의 기정망마스크보다 더 짧을수 있다(routing(7)을 참고).

만일 선택항목 netmask가 주어 지지 않으면 경로의 마스크는 국부대면부와 관련된 망마스크로부터 유도될것이다(ifconfig(1)을 참고).

마스크는 같은 망주소들을 가지는 이 국부대면부의 가장 긴 망마스크로 기정화될것이다. 만일 같은 망주소를 가지는 임의의 국부대면부가 없다면 마스크는 목적지의 기정망마스크를 기정화될것이다.

getway 목적지가 도달할 관문. 관문은 다음과 같은것들중의 하나이다.

- 호스트이름(공식이름 혹은 별명. gethostbyname(3N)을 참고)
- 점표시에서 인터네트주소(inet(3N)을 참고)

count 관문이 원격호스트 혹은 국부호스트인가를 지적하는 옹근수이다. 만일 경로가 원격관문을 통하여 목적지에 도달한다면 계수기는 0보다 큰 수로 되여야 한다. 만일 경로가 목적지에 이르고 관문이 국부호스트이면 계수기는 0으로 되여야 한다. 계수기의 기정값은 0이다. 만일 계수기가 부수값을 가진다면 결과는 정의되지 않는다.

연산

목적지 혹은 관문으로 규정된 모든 기호적이름은 gethostby name()를 리용하여 첫번째 호스트이름으로 볼수 있어야 한다. 만일 호스트이름이 찾아 지지 않는다

면 목적지는 getnetbyname()를 리용하며 망이름으로서 찾을수 있다. 목적지와 관문은 점표시로 되여야 한다(inet(3N)을 참고).

만일 선택항목 -n 가 규정되지 않으면 임의의 호스트와 망주소는(기정으로 인쇄되는) 기정망주소와 알려 지지 않은 이름을 가진 주소를 제외하고 각각 gethostbyname()와 getnetbyname()으로 귀환된 이름에 따라 기호적으로 표시된다. 알려 지지 않은 이름을 가진 주소는 인터네트점표시로 인쇄된다(inet(3N) $>를 참고).

만일 선택항목 - n 이 규정되면 임의의 호스트와 망주소는 기정으로 인쇄되는 기정망주소를 제외하고 인터네트점표시로 인쇄된다.

만일 선택항목 -f 가 규정되면 경로는 관문의 원격호스트를 규정하는 모든 경로표의 등록항목들을 제거한다. 만일 우에서 서술된 부분지령들중의 어느한 지령이 리용된다면 등록항목들은 부분지령이 수행되기전에 제거된다.

경로에서 MTU 의 발견은 중간경로에서 자료기입을 토막화하지 않고 인터네트경로에 보낼수 있는 IP 자료기입의 최대크기를 알기 위한 기술이다. 본질적으로 이 기술을 봉사하는 원천호스트는 나가는 대면부의 크기까지 자료기입을 시동순간에 보낸다. IP 자료기입의 머리부에서 토막(DF)비트는 설정되지 않는다. 경로에서 MTU 의 발견을 봉사하는 중간경로조종기는 다음 단계의 경로조종기에 한 쪼각으로 전송하기에는 너무 큰것으로서 DF 비트가 설정된 자료기입을 받는다. 그다음 경로조종기는 지료기입을 분산시키고 필요되는 토막화와 DF 설정을 의미하는 코드로 ICMP 목적지의 도달불가능에 대한 정보를 보낸다. ICMP 정보는 다음단계의 경로조종기의 MTU 를 포함할것이다. 원천호스트가 ICMP 의 정보를 받을때 그것은 ICMP 의 정보에서 MTU 에로 경로의 MTU 를 축소시킨다. 이 기술을 리용할 때 이 경로의 원천호스트에서 호스트경로는 보통 MTU 를 포함할것이다.

기정으로 경로에서 MTU 의 발견은 TCP 소케트들에서는 가능하고 UDP 소케트들에서는 불가능하다.

만일 선택항목 -p pmtu 가 호스트경로에서 규정된다면 pmtu 의 값은 호스트경로에서 변경되지 않고 리용된다. 만일 경로에서 MTU 의 발견처리가 후에 이 경로에서 보다 더 작은 pmtu 를 발견한다면 호스트경로에서 마당 pmtu 은 갱신되지 않는다. 예고통보는 새 pmtu 의 값으로 체계작업일지에 기입될것이다.
선택항목 -p pmtu 는 호스트경로에 있는 적당한 pmtu 로 충분한 망환경이 알려질 때에만 유용할것이다. IP 는 원격호스트에로 자료기입을 보내기전에 국부호스트의 경로로 규정된 pmtu 로 자료기입을 구분할것이다. 그것은 만일 지령 route 에서 규정된 pmtu 가 정확하다면 경로를 따라서 경로조종기에 의하여 토막화를 피하게 한다.

ping 은 원격호스트의 경로의 정보 pmtu 를 찾는데 리용될수 있다. 경로화표에서 정보 pmtu 는 지령 netstat -r 로 표시될수 있다(netstat(1)을 참고).

출력

add destination:getway getway

규정된 경로는 표들에 추가되였다.

delete destination:getway getway

규정된 경로는 표들로부터 제거되였다.

기발들

지령 route 에서 계수기와 목적지형마당들의 값은 netstat -r 표시에서 G 와 H 기발의 규정으로 결정되며 다음표에서 보여 준것처럼 경로형을 표시한다.

계수기	목적지형	기발들	경 로 형
=0	망	U	국부호스트로부터 직접 망까지 경로
>0	망	UG	원격호스트관문을 통하여 망까지 경로
=0	호스트	UH	국부호스트로부터 직접 원격호스트까지 경로
>0	호스트	UGH	원격호스트의 관문을 통하여 원격호스트까지 경로
=0	기정	U	국부호스트로부터 직접 그룹의 경로
>0	기정	UG	원격호스트관문을 통하여 그룹의 경로

진단들

다음 오유진단들이 표시될수 있다.

add a route that already exists

규정된 등록항목은 경로화표에 이미 있다.

add too many routes

경로화표는 초과되여 있다.

delete a roate that does not exist

규정된 경로는 경로화표에 없다.

경고

호상간 경로지령들은 국부호스트, 목적지호스트, 만일 경로화가 가상회로접속 혹은 비직접자료기입전송의 경우에 성공한다면 모든 중간호스트들에서 수행되여야 한다.

HP−UX 에서 경로의 실현은 change 의 부분지령을 현재 봉사하지 않는다.

저자

route 는 캘리포니아종합대학, 버클리에 의해서 개발되였다.

파일들

/etc/networks
/etc/hosts

관련항목

netstat(1), ifconfig(1M), ping(1M), getsockopt(2), recv(2), send(2), gethostbyname(3N), gethostbyaddr(3N), getnetbyaddr(3N), getnetbyname(3N), inet(3N), roating(7)

rpcinfo

rpcinfo － **원격틀호출**(RPC - Remote Procedure Call)정보를 통보한다.

---

rpcinfo(1M)

이름

rpcinfo － RPC 정보를 통보한다.

형식

/usr/sbin/rpcinfo -p [host]
/usr/sbin/rpcinfo [-n portnum] -u host program [version]
/usr/sbin/rpcinfo [-n portnum] -t host program [version]
/usr/sbin/rpcinfo -b program version
/usr/sbin/rpcinfo -d program version

해설

rpcinfo 는 봉사기 RPC 에 RPC 를 호출하고 그것이 무엇을 하였는가를 통보한다.

선택항목들

rpcinfo 는 다음과 같은 지령행선택항목들과 인수들을 가진다.

-p host　　호스트에서 **포구변환자**(Portmapper)를 엄밀히 조사하고 모든 동등한 RPC 프로그람의 목록을 인쇄한다. 만일 호스트가 규정되지 않으면 호스트이름으로 귀환된 값을 기정으로 리용한다 (hostname(1)을 참고).

-n portnum　　포구변환자로 주어 진 포구번호대신에 선택항목 -t 와 -u 의 포구번호와 같은 포구번호를 리용한다.

-u　　UDP 를 리용하여 규정된 호스트에서 프로그람의 틀 0 의 RPC 를 호출하고 어떤 응답을 받았는가를 통보한다.

-t	TCP 의 리용으로 규정된 호스트에서 프로그람의 틀 0 의 RPC 를 호출하고 어떤 응답을 받았는가를 통보한다.
-b	UDP 의 리용으로 규정된 프로그람과 방안의 틀 0 의 RPC 의 방송(공개)을 하고 응답하는 모든 호스트들에 대한 통보를 한다.
-d	규정된 프로그람과 방안의 RPC 봉사의 등록을 제거한다. 적당한 특정권을 가진 사용자만이 이 선택항목을 리용할수 있다.
program	이름 혹은 수값으로 될수 있다.
version	만일 이 인수가 규정된다면 rpcinfo 는 규정된 프로그람의 그 방안을 호출하려고 한다. 다른 경우에 rpcinfo 는 방안 0 호출로 규정된 프로그람의 모든 등록된 판본번호를 찾으려고 시도한다. 그리고 다음에 매 등록된 방안의 호출을 시도한다(방안 0 은 존재하지 않는것으로 지적되여 있으나 실지로 방안 0 이 있다면 rpcinfo 는 대신에 가장 높은 판본번호의 호출로 판본번호의 정보를 얻으려고 한다.). 선택항목 -b 와 -d 가 리용될 때 version 은 규정되여야 한다.

실례들

국부기계(Local Machine)에 등록된 모든 RPC 봉사를 보여 준다.

```
rpcinfo -p
```

이름 klaxon 을 가진 기계에 등록된 모든 RPC 봉사를 보여 준다.

```
rpcinfo -p klaxon
```

망정보봉사(NIS)를 수행하는 국부망에서의 모든 기계들을 보여 준다.

```
rpcinfo -b ypsorv 1 | sort | unig
```

여기서 1 은 앞선 실례에서 선택항목 -p 의 결과로부터 얻은 현재 NIS 방안이다.
walld 봉사의 방안 1 의 등록을 제거한다.

```
rpcinfo -d walld 1
```

(walld 는 rwalld 의 RPC 프로그람이름이다(rwalld(1M)을 참고).

경고

Sun UNIX 3.0 보다 이전 방안에서 망의 파일체계(NFS)는 포구변환자로 등록되지 않는다. rpcinfo 는 이전 방안으로 수행되는 호스트들에 있는 NFS 봉사기에 대한 RPC 호출을 리용할수 없다. NFS 의 임의의 HP 방안에서 이것은 적용되지 않는다.

저자

rpcinfo 는 Sun Microsystems, Inc 에 의하여 개발되였다.

파일들

/etc/rpc RPC 프로그람번호들의 이름

관련항목

rpc(4), portmap(1M)
NFS 봉사를 위한 프로그람화와 규약들.

rwho

rwho – 원격체계에서 사용자들의 목록을 표시하여 준다.

---

rwho(1)

이름

rwho – 국부기계에 가입된 사용자들을 보여 준다.

형식

rwho [-a]

해설

rwho 는 rwho 데몬인 rwhod 를 수행할 때 국부망에서의 모든 기계들에 지시하는 HP-UX 의 출력과 류사한 출력을 생성한다(who(1)과 rwhod(1M)을 참고). 만일 rwhod 가 11 분동안 한 기계로부터 통보를 받지 못하면 rwho 는 기계가 내려 진것으로 가정하고 그 기계에 가입된 마지막사용자들을 통보하지 않는다.
rwho 의 출력행들은 사용자이름, 기계이름, 사용자의 말단형, 가입한 사용자시간, 놀고 있는(idle) 사용자시간의 총계를 표시하는 마당들을 포함한다. 놀고 있는 시간은

hours:minutes

형식으로 보여 준다.
만일 사용자가 1 분이상 체계에 아무것도 입력하지 않으면 rwho 는 놀고 있는 시간으로서 이것을 통보한다. 만일 사용자가 한시간이상 체계에 입력하지 않고 기발 -a 가 주어 지지 않는다면 사용자는 rwho 의 출력으로부터 생략된다.

실례로 rwho 의 출력형식은 다음의 행과 류사하다.

joe_user machine1: ttyop1 sep 12 13:28 :11

이 출력행은 joe_user 가 기계에 가입되고 그 말단행은 ttyop1 이라고 해석된다. joe_user 는 9 월 12 일 13h 28min 에 가입하였고(1:28 p.m.) 11min 동안 기계에 아무것도 입력하지 않았다는것을 보여 준다.

경고

rwho 의 출력은 rwhod 를 수행하는 국부망의 매 기계의 사용자의 수가 많을 때 부피가 지나치게 크게 된다. 출력의 한 행은 rwhod 를 수행하는 국부망의 매 기계에서 매 사용자에 대하여 나타난다.

저자

rwho 는 캘리포니아종합대학, 버클리에 의해서 개발되였다.

파일들

/var/spool/rwho/whod.* 다른 기계에 대한 통보

관련항목

ruptime(1), rusers(1), swhod(1M)

telnet

telnet – TELNET 규약을 위한 사용자대면부를 제공한다.

---

telnet(1)

이름

telnet – TELNET 규약을 위한 사용자대면부를 제공한다.

형식

telnet [[options] host [port]]

해설

telnet 는 TELNET 규약을 리용하여 다른 호스트와 통신을 하는데 리용된다. 만일 telnet 가 인수들이 없이 기동된다면 그것은 그 입력재촉문(telnet>)으로 지적되는 지령방식에 들어 간다. 이 방식으로 아래에서 목록화된 지령들을 접수하고 수행시킨다. 만일 telnet 가 인수를 가지고 발동되면 이 인수들로 주어 진 지령들

을 수행시킨다.

일단 접속이 열렸다면 telnet 는 입력방식에 들어 간다. 입력방식은 원격체계의 봉사에 따라서 한번에 한개 기호별로 혹은 행별로 할수 있다.

한번에 한개 기호의 방식에서 입력된 대부분의 본문들은 처리를 위하여 원격호스트로 직접 보낸다.

행별방식에서 모든 본문들은 국부적으로 반사되며(표준으로) 원격호스트에 완성된 행들만 보낸다. 국부반사기호(처음에 ^E)는 국부적반사를 허용, 취소하는데 리용될수 있다. (이것은 대부분 통과암호를 표시하지 않는 통과암호입력에 리용될수 있다.)

다른 방식에서 만일 국부기호의 고정된 인수가 TRUE 이면(행방식에서 기정) 사용자의 기호 guit 와 intr 들은 국부적으로 추적되고 원격의 측면에서 볼 때 TELNET 규약의 렬로서 보낸다. 말단에 순차적출력을 갱신(원격호스트가 TELNET 렬을 리해할 때까지)하고 이전 말단입력을 갱신 guit 와 intr 의 경우에) 하기 위한(아래서 자동갱신프로쎄스와 자동동기프로쎄스를 참고) 선택항목들도 있다.

원격호스트에 접속될동안 telnet 지령방식은 telnet 탈퇴기호(처음에 ^])의 입력방식으로 들어 간다. 지령방식에서 표준말단편집기능을 리용할수 있다.

telnet 는 원격호스트에 봉사기와 통신할 때 8bit 기호를 봉사한다. 8bit 기호를 리용하기 위하여 말단 혹은 원격호스트를 적당히 재모형화하여야 한다. 더 나아가서 telnet 와 원격호스트사이에 8bit 기호를 호출할수 있게 2 진고정인수를 리용해야 한다. 일부 원격호스트들은 8bit 기호의 필요한 봉사를 제공하지 않는다.

임의의 시간에 만일 telnet 가 접속할 때 봉사기에 쓰거나 읽을수 없다면 외부적호스트로 닫긴 접속통보는 표준오유로 출력된다. 다음에 telnet 는 값 1 로 끝난다.

telnet 는 TAC 사용자 ID 선택항목을 봉사한다. (TAC 접근조종체계 혹은 TACACS 사용자 ID 는 미리 알려 져 있다.) 호스트봉사기에서 선택항목의 허용은 사용자가 잠간동안 가입렬을 예고함이 없이 그 호스트에서 telnet 에 가입하게 한다. TAC 사용자 ID 선택항목은 원격호스트들과 사용자들에 대하여 **권한 있는 접근**(Authorizing Access)을 위한 rlogin 과 같은 안전수단을 리용한다. 체계관리자는 관여하는 주콤퓨터들로서 지정된 체계에서만 (telnetd)선택항목을 리용할수 있게 한다. 체계관리자는 그 특징을 리용하도록 TAC 사용자 ID 의 매 사용자에게 매체계의 같은 UID 를 부과해야 한다(telnetd(1M)과 system administration tasks manual(체계관리과제의 편람)PH2355-90051 을 참고).

다음과 같은 telnet 선택항목들을 리용할수 있다.

-8 국부 tty 에서 cs8(8bit 전송)을 허용한다.

-ec telnet 지령방식의 탈퇴기호를 그 기정값 ^]대신에 ^c 로 설정한다.

-l 가입사용자이름과 사용자에게 통파암호에 대한 예고를 하기 위하여 만일 의뢰기에서 가능하다면 TAC 사용자 ID 선택항목은 리용할수 없다. 선택항목 -l 를 생략하면 기정값으로 설정한다.

지령들

다음의 지령들은 지령방식에서 리용될수 있다. 지령을 유일하게 동일시하기 위하여 매 지령에 필요한 형만을 준다(이것은 지령 mod, set, toggle, display 들에 대한 인수들에 대해서도 성립한다.).

open host [port]
지적된 포구에서 host 이름을 가진 호스트와 접속한다. 만일 포구가 규정되지 않으면 telnet 는 표준 TELNET 포구에서 TELNET 봉사기를 조종하려고 시도한다. 호스트이름은 gethostbyname( )으로 읽혀 지는 공식적이름 혹은 별명 혹은 hosts(4)에서 서술한바와 같이 점표시에서 규정된 인터네트주소로 되여야 한다. 만일 호스트이름이 주어 지지 않으면 telnet 는 그것에 대하여 예고한다.

close TELNET 가입기간을 닫는다. 만일 가입기간이 지령방식으로부터 시작된다면 telnet 는 지령방식으로 귀환된다. 다른 경우에 telnet 는 끝난다.

quit 임의의 열린 TELNET 가입기간을 닫고 telnet 는 끝난다. 그리고 (지령방식에서)파일의 끝에 도달하면 가입기간을 닫고 끝낸다.

z telnet 를 연기한다. 만일 telnet 가 일감조종을 봉사하는 쉘로부터 수행된다면 지령 z 은 TELNET 가입기간을 지연하고 telnet 를 기동한 쉘에로 사용자를 귀환한다. 그다음 일감은 지령 fg 로 갱신된다(csh(1)혹은 ksh(1)을 참고).

mode mode
기호(한번에 한개 기호방식) 혹은 행(행별방식)으로 될수 있는 telnet 사용자의 입력방식을 mode 로 변경시킨다. 원격호스트는 요구되는 방식으로 갈 때 허용성을 질문한다. 만일 원격호스트가 그 방식에 들어 갈수 있다면 요구되는 방식에 들어 간다. 기호방식에서 telnet 는 입력된것처럼 원격호스트에 매 기호를 보낸다. 행별방식에서 telnet 는 행으로 사용자입력를 나누고 사용자가 CR, LF 혹은 EOF(표준적으로 ^D)를 입력할 때 원격호스트에 매 행을 전달한다. 행별방식은 국부반사도 설정한다. 사용자입력를 기호별로 해석할것을 요구하는 응용프로그람들(more, csh, ksh, vi 와 같은)은 행별방

식으로 정확히 작업하지 못한다.

status telnet의 현재 상태를 보여 준다. telnet는 현재탈퇴기호를 통보한다. 만일 telnet가 접속되였다면 그것은 접속된 호스트와 현재방식을 통보한다. 만일 telnet가 원격호스트에 접속되지 않았으면 그것은 접속되지 않았다는것을 통보한다. 일단 telnet가 접속된다면 그 지령은 국부적흐름조종의 고정인수값을 통보한다.

display [argument…]
값의 설정과 고정된 모든 혹은 일부 인수값들을 표시한다(아래를 참고).

? [command]
도움말을 얻는다. 인수가 없다면 telnet는 도움말개요를 인쇄한다. 만일 지령이 규정되면 telnet는 그 지령에 대해서만 리용할수 있는 도움말통보를 인쇄한다. 도움말통보는 매 지령당 한개 행씩 서술한다.

! [shell_command]
쉘탈퇴를 규정한다. 환경변수 SHELL은 지령수행에 리용될 쉘의 이름을 검사한다. 만일 shell_command가 규정되지 않으면 shell은 표준으로 사용자말단에 접속된다. 만일 SHELL이 정의되지 않으면 /usr/bin/sh을 리용한다.

send arguments
한개 혹은 더 많은 전문기호렬을 원격호스트에 보낸다. 매 인수들은 다음의 값들중에서 임의의 값을 가질수 있다(다중인수들을 매 send 지령으로 규정할수 있다.).

escape 현재 telnet탈퇴기호(처음에 ^])를 보낸다.

synch TELNET SYNCH 렬을 보낸다. 이 렬은 원격체계에 미리 입력된 모든 입력(아직 읽지 않은)을 버린다. 이 렬은 TCP의 자료를 지체하지 않고 보낸다. (만일 작업하지 않으면 말단에서 반사될수 있는 일부 체계에서 소문자 "r"는 리용할수 없다.)

brk 원격체계에서 의미를 가질수 있는 렬 TELNET BRK(break)을 보낸다.

ip 원격체계가 현재수행프로쎄스의 정지를 일으키는 TELNET IP(프로쎄스중단)를 보낸다.

ao 원격체계로부터 사용자의 말단으로 모든 출력를 갱신하도록 하

는 TELNET AO(출력정지)를 보낸다.

ayt 원격체계가 응답을 선택하거나 선택하지 않는 렬 TELNET AYT을 보낸다.

ec 원격체계가 마지막에 입력된 기호를 지우도록 TELNET EC(지우기기호)를 보낸다.

el 원격체계가 현재 입력된 행을 지우도록 하는 렬 TELNET LC(행지우기)을 보낸다.

ga 원격체계에서 의미를 가지지 않는 렬 TELNET GO 을 보낸다.

nop 렬 TELNET NOP을 보낸다.

? 보낼 지령에 대한 도움말통보를 인쇄한다.

set variable_name value

telnet의 변수들중의 한 변수를 규정된 값으로 설정한다. 규정된 값 off는 변수와 관련된 기능을 취소한다. 변수의 값은 지령 display를 리용하여 볼수 있다. variable_name은 다음과 같은 값들중의 어느한 값으로 규정될수 있다.

echo 행별방식에서 입력된 기호의 론리적반사(표준처리)와 입력된 기호들을 금지하는 반사(실례로 통과암호입력에)중의 어느 한 고정인수의 값(시작에서 ^E)이다.

escape 이것은 telnet 지령방식에로 입력하게 하는 telnet 탈퇴기호(시작에서 ^])이다(원격체계에 접속될 때).

interrupt 만일 telnet가 국부기호방식에 있고(아래에 제시된 국부기호 고정인수들을 참고) 중단기호가 입력되면 렬 TELNET IP를 원격호스트에 보낸다. 중단기호의 시동값은 말단기호 intr로 취한다.

quit 만일 telnet가 국부기호방식에 있고 **탈퇴기호**(Quit Character)가 입력되면 TELNET BRK 렬을 원격호스트에 보낸다. 탈퇴기호의 시동값은 말단의 탈퇴기호로 취한다.

flushoutport

만일 telnet가 국부기호방식에 있고 출력취소기호가 입력되면 렬 TELNET AO을 원격호스트에 보낸다. 취소기호의 시동값은 ^O이다.

erase 만일 telnet가 국부기호방식에 있고 telnet가 한번에 한개 기

호방식으로 처리한다면 이 기호가 입력될 때 렬 TELNET EC 을 원격체계에 보낸다. **지우기기호**(Erase Character)의 시동값을 말단의 지우기기호로 취한다.

kill　　만일 telnet 가 국부기호방식에 있고 telnet 가 한번에 한개 기호방식으로 처리한다면 이 기호가 입력될 때 렬 TELNET EL 을 원격체계에 보낸다. **없애기기호**(Kill Character)의 시동값을 말단의 죽이기기호로 취한다

eof　　만일 telnet 가 행별방식에서 처리된다면 행에서의 첫 기호로 이 기호가 입력될 때 원격체계에 이 기호를 보낸다. 파일끝기호의 시동값을 말단의 파일끝기호로 취한다.

toggle arguments…

telnet 가 사건에 대한 응답을 조종하기 위하여 여러가지 기발들을 고정한다(TRUE 와 FALSE 사이에). 여러개 인수가 규정될수 있다. 이 기발들의 상태는 지령 display 을 리용하여 볼수 있다. 리용할수 있는 인수는 다음과 같다.

localchars

TRUE 이면 취소, 중단, 포기, 지우기, 없애기기호들(각각 a0, ip, brk, ec, el)이 국부적으로 식별되며 적당한 TELNET 탈퇴렬로 전송된다. 이 고정인수의 시동값은 행별방식에서는 TRUE 이고 한개 기호방식에서는 FALSE 이다.

autoflush

만일 autoflush 와 localchars 들이 다 TRUE 이면 기호 a0, intr, quit 들은 항상 식별되고(TELNET 렬로 전송된다.)telnet 는 이 TELNET 렬로 처리된다는 것을 원격체계가 알 때까지 사용자 말단에서 임의의 자료를 표시하기 위하여 다시 리용한다. 이 고정인수의 시동값은 TRUE 이다.

autosynch

만일 autosynch 와 localchars 가 둘다 TRUE 이면 기호 intr 혹은 quit 가 입력될 때 TELNET 렬의 보내기의 결과는 TELNET SYNCH 렬에 뒤따른다. 이 틀은 원격체계가 두개의 TELNET 렬을 읽고 행동할 때까지 이미 입력된 모든 입력에 대한 무시를 시작하도록 한다. 이 고정인수의 시동값은 FALSE 이다.

binary　　두개의 입력과 출력들은 TELNET BINARY 선택항목을 허용

하거나 금지한다. 이 선택항목은 TELNET 봉사기로부터 혹은 이 봉사기에로 8bit 기호의 보내기와 받기를 할수 있게 한다.

crlf　만일 TRUE 이면 행의 끝기호는 ASCII 의 쌍 CR 와 LF 로 보낸다. 만일 FALSE 이면 행의 끝기호렬은 ASCII 의 두 기호 CR 와 NUL 을 보낸다. 이 고정인수의 시동값은 FALSE 이다.

crmod　CR 방식을 고정한다. 이 방식이 가능할 때 원격호스트로부터 받은 임의의 CR 기호들은 CR 와 LF 로 대응시킨다. 이 방식은 사용자에 의하여 입력된 기호들에 작용하지 않는다. 이것은 받기에만 리용된다. 이 방식은 국부반사를 할 의뢰기의 일부 호스트들에서만 요구되지만 출력에서는 CR 를 없앤다. 이 고정인수의 시동값은 FALSE 이다.

echo　국부적반사방식 혹은 원격반사방식을 고정한다. 국부반사방식에서 사용자입력은 원격호스트에 전송되기전에 국부적 telnet 에 의하여 말단에 반사된다. 원격반사방식에서 사용자입력에 대한 임의의 반사는 원격호스트에 의하여 진행된다. c 쉘, korn 쉘, vi 와 같은 그자체로 사용자입력의 반사를 조종하는 응용프로그람들은 정확히 국부반사로 작업할수 없다.

options　처리할 TELNET 선택항목들에 대한 보기를 고정한다. 선택항목들의 보기가 가능할 때 모든 TELNET 선택항목들의 리용상태는 표시된다. telnet 에서 모형선택항목들은 SENT 로 표시되고 선택항목들을 TELNET 로부터 받을동안 RCVD 로 표시된다. 이 고정인수의 시동값은 FALSE 이다.

netdata　모든 망자료에 대한 표시를 고정한다(16 진형식으로). 이 고정인수의 시동값은 FALSE 이다.

?　리용할수 있는 toggle 지령들을 표시한다.

귀환값

오유가 나타나거나 혹은 만일 TELNET 접속이 원격호스트로 닫긴다면 telnet 는 1 값을 귀환한다. 다른 경우에는 0 을 귀환한다.

진단

다음의 진단통보는 telnet 로 표시될수 있다.

telnet/tcp:unknown service

telnet service(4)자료기지에서는 TELNET 봉사등록항목을 찾아볼수 없었다.

hostname:unknown host

telnet 는 호스트이름을 인터네트주소로 대응시킬수 없었다. 다음 걸음은 체계관리자가 호스트들이 자료기지에서 원격호스트의 어떤 등록항목이 있는가를 검사하도록 하는것이다.

? invalid command

틀린 지령이 telnet 지령방식에서 입력되였다.

system call>: …

규정된 체계호출에서 나타난 오유이다(오유의 서술에 대한 적당한 편람등록사항을 참고).

저자

telnet 는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

관련항목

csh(1), ksh(1), login(1), rlogin(1), stty(1), telnets(1M), hosts(4), services(4), termio(7)

telnet(1) - kerbers 인증으로 인터네트봉사를 안전하게 한다.

이름

telnet - TELNET 규약에서 사용자대면부

형식

telnet [[options] host [port]]

해설

telnet 는 TELNET 규약을 리용하여 다른 호스트와 통신을 하는데 리용된다. 만일 telnet 가 인수들이 없이 기동된다면 그것은 그 입력재촉문(telnet>)으로 지적되는 지령방식에 들어 간다. 이 방식으로 아래에서 목록화된 지령들을 접수하고 수행시킨다. 만일 telnet 가 인수를 가지고 발동되면 이 인수들로 주어 진 지령들을 수행시킨다.

일단 접속이 열렸다면 telnet 는 입력방식에 들어 간다. 입력방식은 원격체계의 봉사에 따라서 한번에 한개 기호 혹은 행별로 할수 있다.

한번에 한개 기호의 방식에서 입력된 대부분의 본문들은 처리를 위하여 원격호스트로 직접 보낸다.
행별방식에서 모든 본문들은 국부적으로 반사되며(표준으로) 원격호스트에 완성된 행들만 보낸다. 국부반사기호(처음에 ^E)는 국부적반사를 허용, 취소하는데 리용될수 있다(이것은 대부분 통과암호를 표시하지 않는 통과암호입력에 리용될 수 있다.).

다른 방식에서 만일 국부기호의 고정인수가 TRUE 이면(행별 방식에서 기정) 사용자의 기호 guit 와 intr 들은 국부적으로 추적되고 원격의 측면에서 볼 때 TELNET 규약의 렬로서 보낸다. 말단에 순차적출력을 갱신(원격호스트가 TELNET 렬을 리해할 때까지)하고 이전 말단입력을 갱신(guit 와 intr 의 경우에)하기 위한(아래의 자동갱신프로쎄스와 자동동기프로쎄스를 참고) 선택항목들도 있다.

원격호스트에 접속될동안 telnet 지령방식은 telnet 탈퇴기호(처음에_^])의 입력방식으로 들어 간다. 지령방식에서 표준말단편집기능을 리용할수 있다.

telnet 는 원격호스트에 봉사기와 통신할 때 8bit 기호를 봉사한다. 8bit 기호를 리용하기 위하여 말단 혹은 원격호스트를 적당히 재모형화하는것이 필요하다. 더나아가서 telnet 와 원격호스트사이에 8bit 기호를 호출할수 있게 2 진고정인수를 리용해야 한다. 일부 원격호스트들은 8bit 기호의 필요한 봉사를 제공하지 않는다.

임의의 시간에 만일 telnet 가 접속할 때 봉사기에 쓰거나 읽을수 없다면 외부적호스트로 닫긴 접속통보는 표준오유로 출력된다. 다음에 telnet 는 값 1 로 끝난다.

기정으로(혹은 선택항목 -a 와 선택항목 -l 의 리용으로) telnet 의 Kerberos 방안은 Kerberos v5 에 기초한 인증을 봉사하는 의뢰기로서 행동한다. Kerberos 의뢰기로서 telnet 는 사용자가 원격체계에서 접근의 인정과 인증을 하게 한다(Kerberos 인정과 인증의 상세한 내용을 알려면 sis(5)를 참고). 그러나 그것은 직접 검사하거나 혹은 보호된 가입기간을 봉사하지 않을것이다. telnet 는 TAC 사용자 ID 선택항목을 봉사한다. (TAC 접근조종체계 혹은 TACACS 사용자 ID 는 미리 알려 져 있다.) 호스트봉사기에서 선택항목의 허용은 사용자가 잠간동안 가입렬을 예고함이 없이 그 호스트에서 telnet 에 가입하게 한다. TAC 사용자 ID 선택항목은 원격호스트들과 사용자들에 대하여 **저작권접근**(authorizing access)을 위한 rlogin 과 같은 안전수단을 리용한다. 체계관리자는 관여하는 주콤퓨터들로서 지정된 체계에서만 (telnetd)선택항목을 리용할수 있게 한다. 체계관리자는 그 특징을 리용하도록 TAC 사용자 ID 의 매 사용자에게 매 체계의 같은 UID 를 부과해야 한다(telnetd(1M) 과 system administration tasks manual(체계관리과제의 편람)을 참고).

다음과 같은 telnet 선택항목들을 리용할수 있다.

-8 국부 tty 에서 cs8(8bit 전송)을 허용한다.

-a Kerberos 지대에 자동적가입을 시도하고 TAC 리용과 ID 선택항목을 사용할수 없게 한다. (이것은 기정 가입방식이다.)

인정선택항목의 NAME 부분통신을 통하여 사용자이름을 보낸다. 리용된 이름은 USER 환경변수로 귀환한것과 같은 현재사용자의 이름이다. 만일 이 변수가 정의되지 않았고 사용자이름이 현재사용자 ID 로 들어 오려고 한다면 getpwnvm(3)으로 귀환될것이다. 다른 경우에 그것은 사용자 ID 와 관계된 이름이다.

-e c telnet 지령방식의 탈출기호를 그 기정값 ^]대신에 ^c 로 설정한다.

-l user 규정된 사용자로서 Kerberos 지대에 자동적가입을 시도하고 TAC 사용자 ID 선택항목의 리용을 할수 없게 한다. 규정된 사용자이름은 접근선택항목의 NAME 부분통신으로 보낸다. 선택항목 -l 을 생략하면 기정값으로 설정한다. 현재선택항목 -l 만을 허용한다.

-P Kerberos 인정과 인증의 리용을 불가능하게 한다. 이 선택항목이 규정될 때 통과암호는 읽을수 있는 형식으로 망을 통하여 보낸다.

-f 원격체계에 전송할 **국부신임장**(Local Credential)들을 허용한다. -f 혹은 -F 들중의 하나만 허용한다.

-F 이미 국부환경에서 전송된 임의의 신임장을 포함하는 원격체계에 전송될 국부신임장을 허용한다. -f 혹은 -F 들중의 하나만 허용한다.

지령들

다음의 지령들은 지령방식에서 리용될수 있다. 지령을 유일하게 동일시하기 위하여 매 지령에 필요한 형만을 준다(이것은 지령 mod, set, toggle, display 들에 대한 인수들에 대해서도 성립한다.).

open [-l user] host[port]

지적된 포구에서 host 라는 이름을 가진 호스트와 접속한다. 만일 포구가 규정되지 않으면 telnet 는 표준 TELNET 포구에서 TELNET 봉사기를 조종하려고 시도한다. 호스트이름은 gethostbyname( )으로 읽혀 지는 공식이름 혹은 별명 혹은 hosts(4)에서 서술한바와 같이 점표시에서 규정된 인터네트주소로 되여야 한다. 만일 호스트이름이 주어 지지 않으면 telnet 는 그것에 대하여 예고한다.

선택항목 -l 은 원격체계에 자동적으로 가입할 때 리용할 사용자이름을 규정하는데 리용된다.

이 선택항목의 리용은 TAC 사용자만이 선택항목을 리용할수 없게

한다.

close TELNET 가입기간을 닫는다. 만일 가입기간이 지령방식으로부터 시작된다면 telnet는 지령방식으로 귀환된다. 다른 경우에 telnet는 끝난다.

quit 임의의 열린 TELNET 가입기간을 닫고 telnet는 끝난다. 그리고 (지령방식에서)파일의 끝에 도달하면 가입기간을 닫고 끝낸다.

z telnet를 연기한다. 만일 telnet가 일감조종을 봉사하는 쉘로부터 수행된다면 지령 z은 TELNET 가입기간을 지연하고 telnet를 기동한 쉘에로 사용자를 귀환한다. 그다음 일감은 지령 fg로 갱신된다 (csh(1) 혹은 ksh(1)을 참고).

mode mode

기호(한번에 한개 기호방식) 혹은 행(행별방식)으로 될수 있는 telnet 사용자의 입력방식을 mode로 변경시킨다. 원격호스트는 요구되는 방식으로 갈 때 허용성을 질문한다. 만일 원격호스트가 그 방식에 들어 갈수 있다면 요구되는 방식에 들어 간다. 기호방식에서 telnet는 입력된것처럼 원격호스트에 매 기호를 보낸다. 행별방식에서 telnet는 행으로 사용자입력를 나누고 사용자가 CR, LF 혹은 EOF(표준적으로 ^D)를 입력할 때 원격호스트에 매 행을 전달한다. 행별방식은 국부반사도 설정한다. 사용자입력를 기호별로 해석할것을 요구하는 응용프로그람들(more, csh, ksh, vi와 같은)은 행별방식으로 정확히 작업하지 못한다.

status telnet의 현재 상태를 보여 준다. telnet는 현재탈퇴기호를 통보한다. 만일 telnet가 접속되였다면 그것은 접속된 호스트와 현재방식을 통보한다. 만일 telnet가 원격호스트에 접속되지 않았으면 그것은 접속되지 않았다는것을 통보한다. 일단 telnet가 접속된다면 그 지령은 국부적흐름조종의 고정인수값을 통보한다.

display [argument…]

값의 설정과 고정된 모든 혹은 일부 인수값들을 표시한다(아래를 참고).

? [command]

도움말을 얻는다. 인수가 없다면 telnet는 도움말개요를 인쇄한다. 만일 지령이 규정되면 telnet는 그 지령에 대해서만 리용할수 있는 도움말통보를 인쇄한다. 도움말통보는 매 지령당 한개 행씩 서술한다.

! [shell_command]

쉘탈퇴를 규정한다. 환경변수 SHELL은 지령수행에 리용될 쉘의 이름을 검사한다. 만일 shell_command가 규정되지 않으면 shell은 표준으로 사용자말단에 접속된다. 만일 SHELL이 정의되지 않으면 /usr/bin/sh을 리용한다.

send arguments

한개 혹은 여러개 전문기호렬을 원격호스트에 보낸다. 매 인수들은 다음의 값들중에서 임의의 값을 가질수 있다. (다중인수들을 매 send지령으로 규정할수 있다.)

escape　현재 telnet 탈퇴기호(처음에 ^])를 보낸다.

synch　TELNET SYNCH 렬을 보낸다. 이 렬은 원격체계에 미리 입력된 모든 입력(아직 읽지 않은)을 버린다. 이 렬은 TCP 의 자료를 지체하지 않고 보낸다. (만일 작업하지 않으면 말단에서 반사될수 있는 일부 체계에서 소문자 "r"는 리용할수 없다.)

brk　원격체계에서 의미를 가질수 있는 렬 TELNET BRK(break)을 보낸다.

ip　원격체계가 현재수행프로쎄스의 정지를 일으키는 TELNET IP(프로쎄스중단)를 보낸다.

ao　원격체계로부터 사용자의 말단으로 모든 출력를 갱신하도록 하는 TELNET AO(출력정지)를 보낸다.

ayt　원격체계가 응답을 선택하거나 선택하지 않는 렬 TELNET AYT을 보낸다.

ec　원격체계가 마지막에 입력된 기호를 지우도록 TELNET EC(지우기기호)를 보낸다.

el　원격체계가 현재 입력된 행을 지우도록 하는 렬 TELNET LC(행지우기)를 보낸다.

ga　원격체계에서 의미를 가지지 않는 렬 TELNET GO 를 보낸다.

nop　렬 TELNET NOP을 보낸다.

?　보낼 지령에 대한 도움말통보를 인쇄한다.

set variable_name value

telnet 의 변수들중의 한 변수를 규정된 값으로 설정한다. 규정된 값 off 는 변수와 관련된 기능을 취소한다. 변수의 값은 지령 display 를 리용하여 볼수 있다. variable_name 은 다음과 같은 값들중의 어느한 값으로 규정될수 있다.

echo　행별방식에서 입력된 기호의 론리적반사(표준처리)와 입력된 기호들을 금지하는 반사(실례로 통과암호입력에)중의 어느 한 고정인수의 값(시작에서 ^E)이다.

escape　이것은 telnet 지령방식에로 입력하게 하는 telnet 탈퇴기호(시

작에서 ^))이다. (원격체계에 접속될 때)

intarrupt 만일 telnet 가 국부기호방식에 있고(아래에서 보여 준 국부기호고정인수들을 참고) 중단기호가 입력되면 렬 TELNET IP 를 원격호스트에 보낸다. 중단기호의 시동값은 말단기호 intr 로 취한다.

quit 만일 telnet 가 국부기호방식에 있고 **탈퇴기호**(quit character)가 입력되면 TELNET BRK 렬을 원격호스트에 보낸다. 탈퇴기호의 시동값은 말단의 탈퇴기호로 취한다.

flushoutport

만일 telnet 가 국부기호방식에 있고 출력취소기호가 입력되면 렬 TELNET AO 을 원격호스트에 보낸다. 취소기호의 시동값은 ^O 이다.

erase 만일 telnet 가 국부기호방식에 있고 telnet 가 한번에 한개 기호방식으로 처리한다면 이 기호가 입력될 때 렬 TELNET EC 을 원격체계에 보낸다. 지우기기호의 시동값을 말단의 지우기기호로 취한다.

kill 만일 telnet 가 국부기호방식에 있고 telnet 가 한번에 한개 기호방식으로 처리한다면 이 기호가 입력될 때 렬 TELNET EL 을 원격체계에 보낸다. 없애기기호의 시동값을 말단의 없애기기호로 취한다.

eof 만일 telnet 가 행별방식에서 처리된다면 행에서의 첫 기호로 이 기호가 입력될 때 원격체계에 이 기호를 보낸다. 파일끝기호의 시동값을 말단의 파일끝기호로 취한다.

toggle arguments…

telnet 가 사건에 대한 응답을 조종하기 위하여 여러가지 기발들을 고정한다(TRUE 와 FALSE 사이에). 여러개 인수가 규정될수 있다. 이 기발들의 상태는 지령 display 을 리용하여 볼수 있다. 리용할수 있는 인수는 다음과 같다.

localchars

TRUE 이면 취소, 중단, 포기, 지우기, 없애기기호들(각각 a0, ip, brk, ec, el)이 국부적으로 식별되며 적당한 TELNET 탈퇴렬로 전송된다. 이 고정인수의 시동값은 행별방식에서는 TRUE 이고 한개기호방식에서는 FALSE 이다.

autoflush

만일 autoflush 와 localchars 들이 다 TRUE 이면 기호 a0, intr, quit 들은 항상 식별되고(TELNET 렬로 전송된다.)telnet 는 이

TELNET 렬로 처리된다는 것을 원격체계가 알 때까지 사용자 말단에서 임의의 자료를 표시하기 위하여 다시 리용한다. 이 고정인수의 시동값은 TRUE 이다.

autologin Kerberos 지대에로의 자동적가입을 가능하거나 불가능하게 한다. 이 선택항목들의 리용은 선택항목 -a 리용과 같은 결과를 가져 온다. 이 고정인수의 시동값은 TRUE 이다.

autosynch

만일 autosynch 와 localchars 가 둘다 TRUE 이면 기호 intr 나 quit 가 입력될 때 TELNET 렬의 보내기의 결과는 TELNET SYNCH 렬에 뒤따른다. 이 틀은 원격체계가 두개의 TELNET 렬을 읽고 행동할 때까지 이미 입력된 모든 입력에 대한 무시를 시작하도록 한다. 이 고정인수의 시동값은 FALSE 이다.

binary 두개의 입력와 출력들은 선택항목 TELNET BINARY 를 허용하거나 금지한다. 이 선택항목은 TELNET 봉사기로부터 혹은 이 봉사기에로 8bit 기호의 보내기와 받기를 할수 있게 한다.

crlf 만일 TRUE 이면 행의 끝기호는 ASCII 의 쌍 CR 와 LF 로 보낸다. 만일 FALSE 이면 행의 끝기호렬은 ASCII 의 두 기호 CR 와 NUL 을 보낸다. 이 고정인수의 시동값은 FALSE 이다.

crmod CR 방식을 고정한다. 이 방식이 가능할 때 원격호스트로부터 받은 임의의 CR 기호들은 CR 와 LF 로 대응시킨다. 이 방식은 사용자에 의하여 입력된 기호들에 작용하지 않는다. 이것은 받기에만 리용된다. 이 방식은 국부반사를 할 의뢰기의 일부 호스트들에서만 요구되지만 출력에서는 CR 를 없앤다. 이 고정인수의 시동값은 FALSE 이다.

echo 국부적반사방식 혹은 원격반사방식을 고정한다. 국부반사방식에서 사용자입력은 원격호스트에 전송되기전에 국부적 telnet 에 의하여 말단에 반사된다. 원격반사방식에서 사용자입력에 대한 임의의 반사는 원격호스트에 의하여 진행된다. c 쉘, korn 쉘, vi 와 같은 그자체로 사용자입력의 반사를 조종하는 응용프로그람들은 정확히 국부반사로 작업할수 없다.

options 처리할 TELNET 선택항목들에 대한 보기를 고정한다. 선택항목들의 보기가 가능할 때 모든 TELNET 선택항목들의 리용상태는 표시된다. telnet 에서 모형선택항목들은 SENT 로 표시되고 선택항목들을 TELNET 로부터 받을동안 RCVD 로 표시된다. 이 고정인수의 시동값은 FALSE 이다.

netdata 모든 망자료에 대한 표시를 고정한다(16 진형식으로). 이 고정인수의 시동값은 FALSE 이다.

?　　　리용할수 있는 지령 toggle 들을 표시한다.

귀환값

오유가 나타나거나 혹은 만일 TELNET 접속이 원격호스트로 닫긴다면 telnet 는 1 값을 귀환한다. 다른 경우에는 0 을 귀환한다.

진단

telnet 로 표시된 진단들은 아래에 표시된다. Kerberos 의 규정오유들은 sis(5)에 목록화되여 있다.

telnet/tcp:unknown service

telnet service(4)자료기지에서 TELNET 봉사등록항목을 찾을수 없었다.

hostname:unknown host

telnet 는 호스트이름을 인터네트주소로 대응시킬수 없었다. 다음걸음은 체계관리자가 호스트들이 자료기지에서 원격호스트의 어떤 등록항목이 있는가를 검사하도록 하는것이다.

? invalid command

틀린 지령이 telnet 지령방식에서 입력되였다.

system call>: . . .

규정된 체계호출에서 나타난 오유이다. 오유의 서술에 대한 적당한 편람등록사항을 참고.

저자

telnet 는 캘리포니아종합대학, 버클리에 의하여 개발되였다.

관련항목

csh(1), ksh(1), login(1), rlogin(1), stty(1), telnets(1M), hosts(4), services(4), termio(7), sis(5)

telnet 를 수행시키기 위하여 telnet 데몬인 telnetd 를 먼저 시작시킬 필요가 있다.

만일 telnetd 가 수행되지 않는다면 시동지시에 대한 temnet 의 편람을 보아야 한다.

telnet 가 가입하여 있을 동안 일반적인 통신교환을 설정하기 위하여 telnetd 는 telnet 가입기간의 의뢰기에 선택항목들을 보낸다.

제１７장. 쏘프트웨어개발의 기초

UNIX 는 높은 성능과 신뢰성, 계량성을 가지는 세련되고 강력한 조작환경이다. UNIX 는 초기에 연구사들에 의하여 개발되였으며 지금은 사무지향의 프로그람들까지 포함하여 많은 응용프로그람에 대한 개발환경으로 발전하였다.

앞 장에서 본바와 같이 UNIX 체계는 수백개의 봉사프로그람들을 포함하는데 이 봉사프로그람들은 지령으로 참조할수 있다. 이런 지령들이 **스크립트**(Script)형식으로 결합되면 그 스크립트는 주어 진 문제를 풀수 있는 프로그람으로 된다.

그러나 스크립트를 작성하는 언어들은 여러가지 부족점을 가지고 있다. 이 언어는 유연하고 사용하기가 쉽지만 콤퓨터의 주기억과 I/O 장치를 직접 관리하지는 못한다. 스크립트작성언어들도 역시 **해석되는 언어**(Interpreted Language)이다. 스크립트에 씌여 진 지령들은 그 스크립트가 수행될 때에만 읽고 평가될수 있다. 즉 스크립트가 수행될 때마다 지령들이 계속 해석되여야 하기때문에 능률이 좋지 못하다.

쉘프로그람을 위한 지령스크립트들은 어느정도 충분하지만 보다 복잡한 문제를 풀려면 프로그람작성언어가 필요하게 된다.

다음의 장들에서는 프로그람작성의 기초와 가장 기본적인 세개의 대중언어들인 C, C++, Java 에 대하여 취급되는데 그것들은 프로그람작성에 대한 기본개념만을 가진 초학자들을 위한것이다. 그 장들에서는 전문프로그람작성언어에 대한 참고서로서의 충분한 정보가 아니라 단순히 개념만 주고 있다. 보다 충분한 정보는 해당한 언어에 대한 참고서들로부터 얻을수 있다.

콤퓨터프로그람에 대한 리해

콤퓨터프로그람(Computer Programs)은 어떤 과제를 수행하기 위하여 혹은 어떤 문제를 풀기 위하여 콤퓨터에게 주는 명령들이다. 사람이 읽을수 있는 명령들을 **원천코드**(Source Code)라고 부른다.

유감스럽게도 콤퓨터는 **기계어**(Machine Language)만을 리해한다. 기계어는 2 진수들의 집합으로서 아무런 지능도 없다. 기계어의 2 진수들은 콤퓨터주기억에 있는 특수한 기계명령들과 주소(위치)에 직접 대응되며 개발자들은 이러한 2 진수들로 프로그람을 작성하여야 한다. 기계어의 한 토막을 보면 다음과 같은 형식으로 되여 있다:

```
00000010101111001010
00000010111111001000
00000011001110101000
```

우의 형식은 아래의 대응하는 현대프로그람작성언어의 원천코드에서와 같이 명백히 안겨 오지 않는다 :

k = i + j ;

언어는 초기에 읽기 쉽고 사용하기 쉽도록 발전시키는것이 필요하였다.

그 다음에 나온 언어가 **아셈블리어**(Assembly Language)이다. 아셈블리어에서는 2진수들의 렬대신에 여러가지 연산을 수행하는 기호적이름들을 사용하고 특수한 기억주소들을 참조하도록 하고 있다. 기계어프로그람안에 들어 있는 2진수들의 렬이 콤퓨터에게 어떤 수를 기억시킬것을 알려 준다면 여기에 동등한 기호적식은 store x 라고 쓸수 있다.

기계어와 아셈블리어를 **저수준언어**(Low - Level Language)라고 부른다. 저수준언어는 시간랑비가 많고 사용하기가 불편하다. 실례로 두개의 수를 더하는것과 같은 단순한 프로그람연산도 여러개의 저수준연산들을 요구한다. 즉 우의 실례 k = i + j ;에 대하여 아셈블리어에서는 다음과 같은 연산들을 요구한다:

load i, add j, store k

여기서 매 걸음들은 프로그람작성자들이 써야 할 명백한 명령문들을 요구한다.

아셈블리어명령문들과 특수한 기계명령들사이에는 1대 1대응관계가 있다.

기계명령모임은 콤퓨터체계에 따라 완전히 다르다. 따라서 프로그람작성자들은 매 콤퓨터체계에서 리용되는 기계명령모임에 대하여 학습하여야 한다.

저수준언어는 호환성이 없는 프로그람을 창조한다. 이것은 기계명령모임에서의 차이로 인하여 저수준언어로 작성된 프로그람이 다른 콤퓨터체계에서 다시 쓰지 않고는 수행될수 없다는것을 의미한다. 아셈블리어프로그람은 이러한 명령모임에 의하여 씌여 지기때문에 그것들은 기계종속성을 가진다.

한편 **프로그람작성언어**(Programming Language)들은 **고수준언어**(Higher – Level Language)로 고찰된다. C, C++, Java 는 프로그람작성언어의 실례로 되며 이러한 언어들은 사람들이 읽을수 있는 언어들이다. 저수준언어에서의 여러 단계들은 고수준언어에서의 하나의 명령문으로 대응시킬수 있으므로 프로그람작성시간, 노력, 난점들이 감소되게 된다. UNIX 도 초기에는 아셈블리어로 작성되였지만 후에 다시 C 언어로 작성되였다. 고수준언어의 문법은 여러 콤퓨터체계들에서 표준화되였다. 따라서 프로그람을 특별한 콤퓨터체계의 명령모임과 련관시킬 필요가 더는 없게 되였다. 프로그람작성자들은 기계에 종속되지 않는 프로그람들을 원천코드로 작성할수 있으며 결국 고수준언어에 의한 프로그람작성은 호환성이 허용되게 되였다. 콤퓨터체계는 원천코드를 리해할수 없게 되여 있다. 원천코드는 기계어로 전환되여야 한다. 이 문제를 해결하기 위하여 **콤파일러**(Compiler)라는 특수한 콤퓨터프로그람이 개발되였다.

콤파일되는 언어와 해석되는 언어

원천코드는 콤퓨터가 리해할수 있게 기계어로 전환되여야 한다. 콤파일되는 언어는 콤파일러가 원천코드를 기계어로 변환할것을 요구하며 해석되는 언어는 **해석기**(Interpreter)가 원천코드를 기계어로 변환할것을 요구한다.

콤파일되는 언어와 해석되는 언어사이의 주되는 차이는 그 언어들이 기계어로 전환될 때에 나타난다.

Java 또는 UNIX 스크립트프로그람과 같이 해석되는 언어는 실행될 때 기계어로 바꾼다. 즉 프로그람이 수행될 때마다 바꾸기가 진행된다. 해석되는 언어들은 콤파일되는 언어보다 천천히 그리고 비효과적으로 실행된다. 그림 17-1 에서 보여 주는것처럼 해석되는 언어에 대하여 기계어수행을 최량화하기 위한 방도는 거의나 없다.

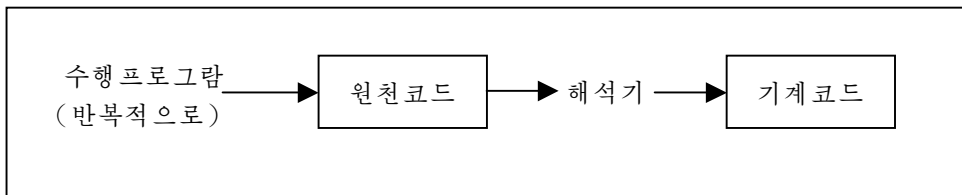



그림 17-1. 해석흐름

한편 C 혹은 C++와 같은 콤파일되는 언어는 수행되기전에 기계어로 바꾸어 진다.

프로그람작성자는 원천코드를 창조한 다음 콤파일러를 실행시키며 그다음 프로그람을 수행시킨다. 따라서 여기서는 콤파일러가 최량화를 진행할수 있는 시간을 얻게 될뿐아니라 기계어로의 바꾸기는 콤파일할 때 한번만 일어 나게 된다. 콤파일은 그림 17-2 에서 보여 주는바와 같이 원천코드에서 다른 변화가 없으면 다시 발생되지 않는다.

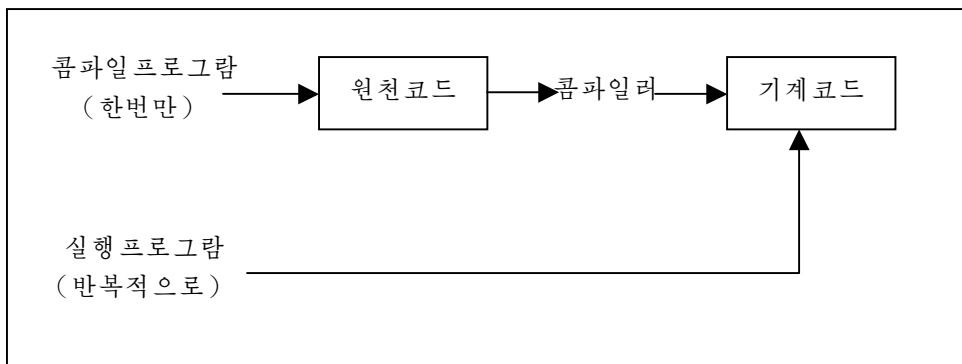



그림 17-2. 콤파일되는 수행흐름

콤파일되는 언어와 해석되는 언어사이의 또 다른 차이는 호환성이 있는 "수행가능한 코드"에서 나타난다.

수행가능한 코드는 프로그람을 수행시키는 파일로서 그 파일이름이 콤퓨터에 입구되면 프로그람이 실행된다. 실례로 C 와 C++에서 수행가능한 코드는 C 와 C++에 대한 기계어로 된다. Java 에 대한 목적언어는 **바이트코드**(Byte Code)이다.

주의 : Java 는 콤파일되며 해석되는 언어이다. 원천코드는 바이트코드로 콤파일되며 바이트코드는 프로그람이 수행될 때 해석된다. 이에 대한 설명은 Java 프로그람이 해석되는 측면을 고찰할 때에 서술한다.

콤파일되는 언어는 콤파일을 수행하는 콤퓨터체계에 대한 콤파일러쏘프트웨어를 요구한다. 콤파일러는 콤퓨터체계의 종류에 따라 특수하게 씌여 진다. 콤파일러에 의하여 생성된 실행가능한 코드는 동일한 기계명령모임을 리용하는 다른 콤퓨터체계에로만 이동될수 있다.

실례로 HP-UX 를 수행시키는 모든 HP 9000 콤퓨터체계들은 개별적인 콤퓨터들에서 다시 콤파일하지 않고도 C 혹은 C++의 실행가능한 코드를 수행시킬수 있다.

만일 C 혹은 C++프로그람이 서로 다른 콤퓨터체계에서 실행되여야 한다면 해당한 콤퓨터체계에 대한 콤파일러가 있어야 하며 원천코드가 재콤파일되여야 한다.

C 와 C++에 대한 수행가능한 코드는 호환성이 없다.

반면에 Java 에 대한 목적언어인 바이트코드는 각이한 콤퓨터체계로 바꿀수 있다. Java 콤파일러에 의하여 콤파일하는 임의의 콤퓨터체계는 표준적인 바이트코드를 생성한다. 바이트코드는 임의의 콤퓨터체계에서 리용할수 있으며 따라서 호환성이 있다.

Java 에서 중요한것은 Java 를 실행시킬수 있는 **가상기계**(Virtual Machine)가 Java 바이트코드를 수행시키는 콤퓨터체계우에 반드시 있어야 한다는것이다. Java 가상기계(JVM)는 수행될 때에 바이트코드를 접수하여 그것을 기계어로 해석한다. JVM 은 기계어로의 전환을 실행한다. 따라서 가상기계는 매 콤퓨터체계에 맞게 작성된다.

실례로 HP 9000 HP-UX 콤퓨터체계에서 발생되는 바이트코드는 재콤파일하지 않고 Sun Solaris 콤퓨터체계에서 실행될수 있다. 이때 Sun 콤퓨터체계는 JVM 을 가지고 있어야 한다. HP 콤퓨터체계에서 리용되는 JVM 은 Sun 콤퓨터체계에서 리용될수 없다.

이처럼 Java 의 바이트코드는 호환성이 있으나 JVM 은 호환성이 없다.

제１８장. 프로그람작성의 구성체

자기가 선택한 프로그람작성언어를 리용하여 프로그람을 창조하려면 우선 해결하여야 할 문제를 리해하여야 한다. 이때 프로그람작성언어에 제한되지 않는 도구들도 필요하게 된다. 론리적도구들을 가지고 있으면 문제해결의 설계뿐아니라 이미 존재하는 프로그람을 리해하는데서 도움을 받을수 있다. 이러한 도구들은 프로그람설계의 론리적흐름을 직관적으로 설명하여 준다.

프로그람작성의 구성체(Programming Construct)는 프로그람의 알고리듬 혹은 론리적 걸음들을 창조하는데 리용될수 있는 론리적도구들이다. 이 장에서는 문제를 프로그람적으로 어떻게 해결하겠는가에 대하여 고찰하며 여러가지 프로그람작성의 구성체들에 대해서도 론의한다.

우선 가장 기본적인 구성체인 대입명령문부터 고찰한다. 프로그람작성의 론리는 문제를 해결하기 위한 추가적인 요소들을 필요로 하기 때문에 수학적인 연산들과 유용한 식들이 도입된다. 대입명령문을 고찰한후에는 선택과 순환의 구성체, 자료구조에 대하여 고찰한다. 이러한 구성체들은 프로그람작성설계에 필요한 론리적인 성분들을 제공하게 된다.

일반형식 혹은 **의사코드**(Pseudocode)를 리용하여 구성체들이 무엇을 하는가에 대한 리해를 가지게 된다.

의사코드는 프로그람안에 들어 있는 프로그람 혹은 명령문들의 식이다. 이러한 의사코드는 콤퓨터언어의 형식으로 되는 간단한 영어문장으로 서술된다. 의사코드는 프로그람을 설계하는 첫 단계에서 가치가 있다. 이 장의 실례들에서는 의사코드를 쓰고 있는데 그것은 프로그람작성언어와 무관계한것이다.

대입구성체

가장 기본적인 프로그람작성의 구성체는 **대입명령문**(Assignment)이다. 수자나 문자와 같은 여러가지 값들이 프로그람안에 있는 변수들에 대입될수 있다. 변수는 프로그람작성자가 준 이름을 가지는 콤퓨터의 주기억안에 있는 어떤 기억장소이다.

변수들은 일반적으로 **자료형**(Data Type)을 배정 받는다. 이 자료형은 변수안에서 취하는 값의 형태가 무엇인가를 가리킨다. 실례로 프로그람작성자는 어떤 변수가 옹근수를 취하도록 할수 있으며 다른 변수는 한개의 문자, 또 다른 변수는 문자렬을 취하도록 할수 있다.

변수는 리용되기전에 선언되여야 한다. 변수선언은 프로그람에서 그 변수가 리용되기전에 한번만 진행되여야 한다. 프로그람을 작성할 때 프로그람의 제일 웃부분에 모든 변수들을 렬거하도록 습관하는것이 좋다. 자료형은 변수가 선언될 때 대입되여야 한다. 다음의 실례에서는 BankAccount 라는 변수가 옹근수를 취하도록 선언되고 있다

는것을 보여 준다. 옹근수는 int 에 의하여 선언된다. 이름 BankAccount 는 프로그람작성자에 의하여 변수에 대입된다. 이에 대한 자세한 내용은 제 22 장과 제 23 장에서 고찰하게 된다.

int BankAccount

변수는 우편함과 같이 생각할수 있다. 우편함은 프로그람작성자가 주는 이름을 배정받는다. 콤퓨터주소는 우편함에 대한 우편주소라고 생각할수 있다. 콤퓨터는 그 주소에 의하여 주기억안에서 우편함이 어디에 있는가를 알게 된다. 우편함안에 있는 우편물은 거기에 대입된 값과 같다.

매개 변수는 프로그람작성자에 의하여 대입된 값을 포함할수 있다. 또한 프로그람이 실행프로쎄스의 변수에 값을 줄수도 있다. 이러한 두가지 작용은 **대입연산**(Assignment Operation)으로 알려 져 있다.

아래의 실례에서는 의사코드에 의한 대입연산을 주고 있다. 대부분의 프로그람작성자들이 "="를 리용하여 대입연산을 표시한다.

BankAccount = 100

우에서 BankAccount 라는 이름이 변수에 대입되였다는것을 보았다. BankAccount 는 물리적으로는 주기억안에 있는 콤퓨터주소이다. 콤퓨터는 콤퓨터주소를 통하여 필요할 때 변수 BankAccount 를 찾을수 있다. 프로그람작성자는 이름 BankAccount 를 참조하게 되며 콤퓨터는 그 이름을 대응하는 콤퓨터주소로 변환한다. 콤퓨터가 주기억에서 그 변수를 찾은후에 변수안에 있는 정보는 프로그람작성자에 의하여 참조, 변경, 제거될수 있다. 우의 실례에서는 프로그람작성자가 BankAccount 라는 이름을 가진 변수에 값 100 을 대입하는것을 보여 주고 있다.

수학적연산자

수학적연산자는 많은 프로그람에서 필요한 부분으로 된다. 프로그람작성언어들은 표준적인 연산자를 표시하는 기호들을 가지고 있다. 수학적연산자들로서는 더하기, 덜기, 곱하기, 나누기, 옹근수나머지, 제곱 등이 있다. 의사코드로 수학적연산자들을 리용할 때 일부 프로그람작성자들은 그 연산에 대한 영어단어를 그대로 쓰는 경우도 있지만 대부분의 사용자들은 짧게 대응하는 기호들을 리용한다.

Add 500 to BankAccount

혹은

BankAccount + 500

프로그람작성언어에서 변수값에 대한 수학적연산자들을 리용하는 문법이 반드시 직관적인것은 아니다. 다음의 실례는 이 문법을 보다 명백히 직관적으로 보여 준다.

BankAccount = BankAccount + 500

일부 프로그람작성자들은 우와 같은 의사코드로 대입명령문을 리용한다. 이 명령문은 BankAccount 변수에 500 을 더 첨가한다는것을 보여 주고 있다. 변수 BankAccount 가 이미 값 100 을 가지고 있었다면 그 값에 500 을 첨가할 때 결과는 600 으로 된다. 즉 값 600 이 BankAccount 에 기호 "="에 의하여 대입된다. 따라서 변수 BankAccount 는 600 이라는 값을 가지게 된다.

비교식

가장 단순한 식은 비교식이다. 비교식은 TRUE 혹은 FALSE 로 평가된다. 비교연산들에는 작기, 크기, 같기, 작거나 같기, 크거나 같기, 론리합, 론리적 등이 있다.

비교식은 식의 두 변의 값들을 계산하고 비교한다. 결과는 TRUE 혹은 FALSE 로 된다. 일반적으로 식을 평가하기 위하여 그것을 명령문처럼 읽는것이 더 좋다. 앞절에서 고찰한바와 같이 아래에서 준 의사코드는 식을 표시하고 있다.

BankAccount < 1000

이 실례에서는 "BankAccount 의 값이 100 보다 작다. "는것을 서술하고 있으며 이 명령문의 결과는 TRUE 혹은 FALSE 로 된다.

만일 BankAccount 의 값이 600 이라면 이 식은 TRUE 로 평가된다. 즉 "600 은 1000 보다 작다."가 옳은 문장으로 평가된다.

만일 BankAccount 의 값이 2000 이라면 이 식은 FALSE 로 평가된다. 즉 "2000 은 1000 보다 작다."는것이 틀린 문장으로 평가된다.

다음의 식을 고찰하자.

PhoneBill >= 50

만일 PhoneBill 의 값이 50 이라면 이 식은 TRUE 로 평가된다. 즉 "50 은 50 보다 크거나 같다."가 옳은 문장으로 된다.

만일 PhoneBill 의 값이 51 이라면 이 식도 역시 TRUE 로 평가된다. 즉 "51 은 50 보다 크거나 같다."가 옳은 문장으로 된다.

그러나 만일 PhoneBill 의 값이 49 라면 이 식은 FALSE 로 평가된다. 즉 "49 는 50 보다 크거나 같다."가 틀린 문장으로 된다.

순환구성체

순환은 그것의 실행회수를 예견할수 있는가 없는가에 따라 두개의 그룹으로 나눌수 있다. 유한반복은 미리 결정된 회수만큼 실행된다. 반면에 무한반복은 프로그람이 실행될 때 그 회수가 결정된다.

유한반복순환의 실례로 1−10 사이의 모든 수자들을 출구하는 프로그람을 들수 있다. 한개의 명령문이 순환부분에 놓이며 10 개의 명령문대신에 그 한개의 명령문이 10 번 실행되여 매 수자가 출구된다.

```
LOOP ten times
    Print number
    Add one to number
```

1 부터 시작하면 유한순환의 결과는 다음과 같다.

1 2 3 4 5 6 7 8 9 10

무한반복순환은 프로그람이 수행되는 프로쎄스에 언제 순환을 정지하겠는가를 결정하게 된다. 10 이하의 수들을 출구하는 while 순환이 바로 무한순환의 한가지 실례이다. 만일 수가 11 이거나 그보다 크다면 순환은 정지되며 순환밖에 있는 다른 명령문이 수행된다.

```
WHILE number is less than or equal to 10
    Print number
    Add one to number
```

만일 다시 1 로 시작하였다면 무한순환의 결과는 다음과 같다.

1 2 3 4 5 6 7 8 9 10

결과가 같다고 해도 여기서 사용한 방법은 다르다. 첫번째 유한순환에서는 1 부터 시작하여 10 번 순환하고 끝낸다. 두번째 무한순환에서는 1 부터 시작하지만 수 10 이 나타날 때까지 몇번 순환하여야 하는지 모른다.

5 로 시작하여 어떤 결과가 나타나는지 살펴 보자.

유한순환에서 5 로 시작하여 10 번 순환하면 결과는 다음과 같다.

5 6 7 8 9 10 11 12 13 14

무한순환에서는 5 로 시작하지만 수가 10 보다 작거나 같을동안에만 순환을 계속한다.

5 6 7 8 9 10

유한순환에서는 순환을 10 번 반복할뿐이지만 이와는 다르게 무한순환에서는 수가 10 보다 작거나 같을동안 순환한다는것이 차이난다.

유한순환은 수 10 이 나타나는가 하는데는 상관이 없으며 그저 10 번 순환하기만 하면 된다. 무한순환에서는 수 10 이 나타날 때까지 5 번 반복하고 순환을 끝낸다.

한개 순환은 다른 순환안에 겹칠수 있다.

겹친순환에 리용되는 공통령역은 표로 고찰될수 있다. 한개 순환은 렬들로 나타나고 다른 순환은 행들로 나타난다.

그림 18-1 에 있는 실례에서 겹친순환은 매 사람들이 작업한 총 시간을 계산하고 있다.

LOOP through three row of employees
LOOP through seven columns of days of the week
Add number of hours worked

	일요일	월요일	화요일	수요일	목요일	금요일	토요일
김영철	8	8	5	8	8	3	0
리순희	0	3	8	8	3	0	0
박웅철	5	0	0	0	3	8	8

그림 18-1. 작업시간표

외부순환은 "김영철"이라는 첫행에서 시작된다. 내부순환은 한개 행에 있는 모든 렬들에 대하여 일요일부터 토요일까지 그 사람의 작업시간을 더하면서 총 40 시간을 얻는 순환이다.

내부순환이 완성되면 조종은 바깥순환으로 거꾸로 전달되여 결국 두번째 행의 "리순희"에게로 가게 된다. 그다음 새로운 내부순환이 시작되여 두번째 행의 모든 렬들에 대하여 일요일부터 토요일까지 그 사람의 작업시간을 추가적으로 더하면서 두사람분의 총 62 시간을 얻는 순환이 진행된다.

또 하나의 내부순환이 완성되면 조종은 바깥순환으로 거꾸로 전달되여 결국 세번째 행의 "박웅철"에게로 가게 된다. 이 내부순환에서는 한주일의 매 요일에 대한 작업시간을 합계하여 결국 모든 사람들의 총 작업시간이 86 시간이라는 결과를 얻게 된다.

선택구성체

선택을 조종하는 론리적구성체들은 프로그람안에서 어떤 결정을 하여야 할 필요가 있을 때에 리용된다. 이 절에서 고찰되는 선택구성체는 if … then …else와 case이다.

if … then …else 명령문

if … then …else 구성체는 식을 평가하여 한가지 선택을 하게 한다. 그 식은 TRUE 혹은 FALSE 이다. 만일 식이 TRUE 라면 then 다음의 명령문을 수행하고 식이 FALSE 이면 else 다음의 명령문을 수행한다.

다음의 실례에서는 잔고가 적어도 $10 이면 $10 만을 회수하는 if … then …else 명령문을 직관적으로 보여 주고 있다.

```
IF bank account balance is equal to or greater than $10
THEN
        Withdraw $10
ELSE
        Print insufficient funds message
```

만일 잔고가 $15 이면 if 명령문은 TRUE 이고 회수가 허용된다.

만일 잔고가 $5 밖에 안된다면 if 명령문은 FALSE 이고 자금이 불충분하다는 통보를 출구하는 else 부분을 수행한다.

else 는 항상 필요한것이 아니다. else 부분이 없고 if 부분이 FALSE 이면 아무것도 수행되지 않으며 프로그람은 계속된다.

다음의 실례를 고찰하자.

```
IF bnak account balance is greater than $200
THEN
        Add interest
```

이 경우에 만일 잔고가 $200 보다 작다면 아무런 리자도 추가되지 않는다. 그리고 프로그람은 if 명령문에서 아무 일도 하지 않고 계속된다.

겹친 if … then … else 명령문

한개의 if … then … else 명령문이 또 다른 if … then … else 명령문안에 겹쳐 질수 있다. 그림 18-2 에서 보여 준것처럼 이러한 내부겹침은 순차적인 선택을 가능하게 할수 있게 한다.

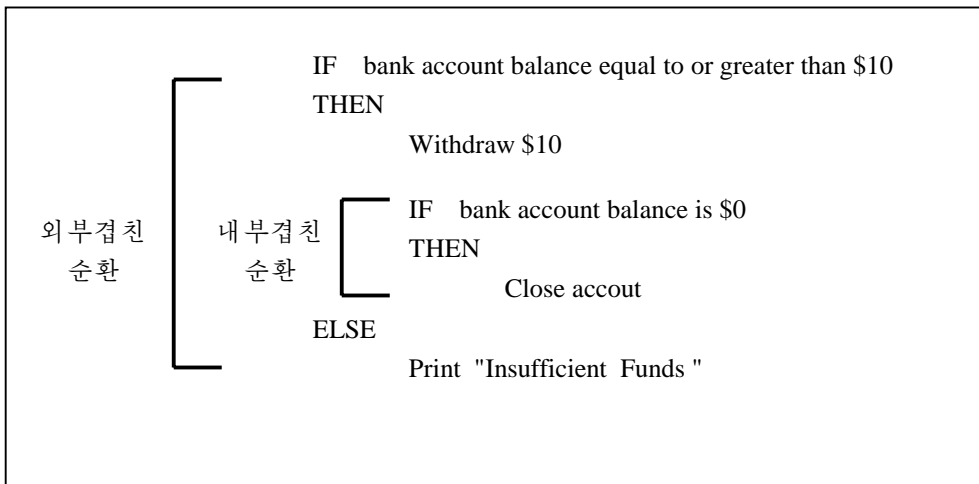



그림 18-2. 겹친 if - then - else

우의 그림에서 ELSE 부분은 외부의 if 명령문에 관계되는데 외부의 THEN 부분안에는 겹친 if … then 부분이 있다. 겹친 if … then 명령문에는 그 명령문과 관련된 ELSE 부분이 없다.

또한 우의 그림에서 if 명령문들을 어떻게 들여 썼는가를 주의해 보아야 한다. 물론 들여쓰기는 프로그람에서 반드시 필요한것은 아니지만 원천코드를 읽는 프로그람작성자들이 혼돈하지 않도록 하기 위하여 들여 쓰는것이 좋다. 그림에서는 외부의 THEN 이 잔고가 $10 이상일 때 수행되며 $10 를 회수하는것을 보여 주고 있다.

겹친 if … then 이 외부 THEN 안에 있기때문에 그것은 다음번에 수행된다. 이 겹친 부분에서는 앞선 회수에 의하여 잔고가 비여 있는가를 검사하고 비여 있다면 잔고계산을 끝낸다. 만일 잔고가 비여 있지 않으면 겹친 if 명령문이 ELSE 부분을 가지지 않기때문에 아무런 작용도 하지 않는다.

프로그람은 외부 및 겹친 if 명령문을 둘 다 수행하고 끝난다.

Case 명령문

Case 명령문은 식의 값을 리용하여 여러개의 가능성들중에서 어느 한개를 선택한다.

```
CASE part number for UNIX software
            101: Print Linux
            102: Print RedHat Linux
            103: Print HP - UX
            104: Print Solaris
ELSE
```

```
            Print bad part number
      END
```

case 명령문은 목록안에 주어 진 번호를 정합하기 위하여 식을 평가한다. 정합이 성공하면 그 부분의 번호를 출구한다. 만일 성공하지 못하면 else 명령문이 수행되며 번호가 틀렸다는 통보가 출구된다.

실례로 번호가 102 인 경우에는 RedHat Linux 가 출구된다. 그러나 번호가 200 이면 번호가 틀렸다는 통보가 출구된다.

자료구조

프로그람에 있는 자료들에 대하여 어떤 일이 발생하겠는가?

프로그람안에 있는 자료 혹은 정보는 혼돈을 일으킬수 있으므로 그것들을 조작하는 일정한 방법이 필요하다. 프로그람언어들은 프로그람작성자가 자료들을 **자료구조**(Data Structure)에 의하여 론리적으로 그룹화하도록 해준다. 프로그람작성자가 자료를 참조하려고 할 때 자료구조의 이름을 리용하도록 할수 있다.

실례로 종업원들과 관리자에 대한 이름과 생활비에 대하여 알고 있을 때 종업원과 관련된 정보는 Employee_Record 라는 자료구조에 넣고 관리자와 관련된 정보는 Manager_Record 라는 자료구조에 넣으면 자료혼돈을 피할수 있다. 이 두개의 자료구조는 다음과 같이 이름과 생활비와 관련한 두개의 객체들을 가진다.

```
STRUCTURE  Employee_Record
                Employee  Name
                Hourly_Pay_Rate
END
STRUCTURE  Manager_Record
                Employee  Name
                Salary_Pay_Rate
END
```

정보를 론리적구조로 조직화하면 프로그람을 읽기도 쉽고 리해하기도 쉽다. 그리고 자료구조의 이름도 가능한껏 설명문을 달아 주는것이 필요하다.

우의 실례를 보면 프로그람작성자들이 어느 정보가 종업원 또는 관리자와 관련되는가를 잘 알수 있다.

이렇게 프로그람이 보다 론리적으로 명백해 지기때문에 종업원생활비와 관리자의 생활비를 혼돈하는 일이 거의나 없게 된다.

제１９장. 프로그람작성설계

프로그람을 작성하기전에 프로그람작성을 계획화할 필요가 있다. 이렇게 먼저 간단히 계획화를 해 놓아야 시간을 절약할수 있으며 후에 실패없이 작업할수 있다.

초학자들에게 있어서 제일 배우기 힘든것은 콤퓨터처럼 사고하는것이다. 이와 관련한 몇가지 기본개념들이 있다.

첫째로, 콤퓨터는 한번에 한 걸음씩 명령을 수행한다. 둘째로, 콤퓨터는 문제를 해결하기 위한 지식을 미리 가지고 있지 않다. 셋째로, 콤퓨터는 모든 일에 대하여 한걸음씩 알려 줄것을 요구하며 사용자가 알려 주는것만을 알아 차린다.

모든 콤퓨터프로그람들은 꼭 같은 일을 수행한다. 프로그람은 콤퓨터에게 자료의 접수(입력), 자료의 조작(처리), 결과의 제시 혹은 거꾸로 사용자에게 정보내보내기(출력)등을 제시한다.

다음과 같은 질문에 대답하면서 론의를 시작하자.

- 콤퓨터로 해결할수 있는 문제들은 어떤것들인가?
- 프로그람으로부터의 출력은 어떻게 보는가?
- 이러한 출력을 달성하는데 필요한 론리적걸음(알고리듬)들은 무엇인가?
- 필요한 입력자료는 무엇인가?

프로그람을 작성할 때 입력, 출력, 론리적걸음들을 정의한후에 마치 콤퓨터가 하는것처럼 문제를 한걸음씩 해결해 나가야 한다.

다음으로 프로그람이 어떻게 조직할것인가를 고찰하여야 한다. 프로그람에 대한 조직의 중요성은 작은 프로그람들에서는 잘 나타나지 않을수 있지만 보다 큰 프로그람에서 그것에 대한 론리적인 리해, 오유수정, 코드의 개선을 하려고 하는 경우에 더 잘 알수 있다.

프로그람작성의 론리는 프로그람에 필요한 한개이상의 기능들을 수행하는 더 작은 론리적쪼각들인 **모듈**(Module)들로 나눌수 있다. 이것은 다른 말로 **분해가능성**(Decomposability)이라고 알려 져 있다.

모듈은 입구를 접수하고 자료를 처리하며 출구를 생성하는 작은 부분프로그람이다. 모듈들은 론리적으로 쉽게 리해되는 순서로 정돈하여 완성된 프로그람을 형성하도록 할수 있다. 이 능력을 **모듈적프로그람작성법**(Modular Programming)이라고 부른다.

모듈적프로그람작성법에 기초한 설계의 한가지 실례를 아래에서 고찰한다. 이 실례에서는 종업원들의 주별 생활비를 계산하는 간단한 프로그람을 보여 주고 있다.

실천적실례

실례를 리용하면 설계의 개념과 걸음을 완전히 리해할수 있다. 아래에 서술된 문제는 상대적으로 쉬운 실례인것처럼 보이지만 들여다 볼수록 일정한 복잡성을 가진다는것

을 알수 있다.

주어 진 문제는 종업원들의 주별 생활비의 총액과 지불액을 계산하는 문제이다. 문제에서는 입구와 기대되는 출구에 대하여 어느 정도 높은 준위의 걸음으로 렬거하고 있다. 이러한 걸음들을 알고리듬으로 보여 준다.

문제서술 : 인체보험금과 세금을 공제하면서 종업원의 주별 생활비의 총액과 지불액을 계산한다.

입력 : 종업원이름
시간당 지불액

출력 : 종업원생활비의 총액와 지불액에 대한 보고서를 출력한다.

알고리듬 :

1. 총액을 계산한다.
2. 지불액을 계산한다.
3. 종업원이름, 총액과 지불액의 보고서를 작성한다.

알고리듬의 매 론리적인 걸음들이 하나 혹은 그것들의 결합으로서 입구를 접수하고 자료를 처리하며 출구를 생성하는 부분프로그람이 될수 있겠는가에 대하여 고찰하여야 한다. 우선 매 론리적걸음에 이름을 달아 주고 필요한 정보(입력)와 자료를 처리한후 결과정보(출력)를 정의하고 그것들이 모듈화될수 있는가를 고찰해 보자 :

1. 모듈 : Calc_Gross_Pay
설명 : 한주일간의 총액을 계산한다.
입력 : 시간당 지불액
출력 : 총액
알고리듬 : 시간당 지불액 * 40

2. 모듈 : Calc_Net_Pay
설명 : 인체보험금을 공제하면서 지불액을 계산한다.
입력 : 총액
출력 : 지불액
알고리듬 : 인체보험금을 계산한다.
총액에서 인체보험금을 공제한다.
세금을 계산한다.
지불액을 계산한다.

3. 모듈 : Generate_Report
설명 : 종업원이름, 총액, 지불액을 생성한다.
입력 : 종업원이름, 총액, 지불액
출력 : 종업원의 주별생활비의 총액과 지불액
알고리듬 : 보고서를 형식화하고 그것을 인쇄한다.

입력, 처리, 출력을 명백히 정의할수 있기때문에 매 알고리듬은 모듈의 요구조건을 만족시킨다.

다음으로 모듈들을 련결하여 하나의 론리적인 흐름을 만든다. 이러한 기능을 **결합가능성**(Composability)이라고 부른다. 그림 19-1 에서 화살표들은 한 모듈로부터의 출구가 다른 모듈의 입구로 된다는것을 보여 주고 있다.

모듈들을 한번 결합하였는데 이제 또다시 그 매개 모듈들을 더 작은 모듈로 나눌수 있겠는가를 살펴 보자.

Calc_Net_Pay 는 인체보험금과 세금을 계산하는 모듈로서 그것의 원천코드는 길고 복잡하며 혼돈될수 있다. 그러므로 이 모듈은 다시 분해될수 있다.

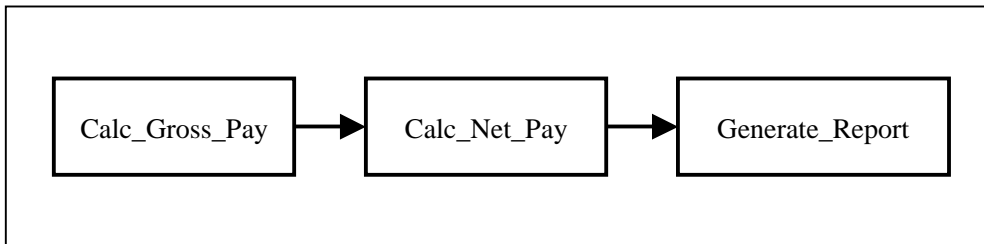



그림 19-1. 모듈정보흐름

그 가능성은 이 모듈을 둘로 나누어 하나는 인체보험금을 계산하고 다른 하나는 세금을 계산하도록 하는것이다. 새 모듈들의 이름은 Calc_Medical 과 Calc_Tax 로 한다.

Calc_Medical 모듈은 Calc_Gross_Pay 모듈로부터 출구되는 정보를 요구한다. 왜냐하면 인체보험금이 총액의 일정한 퍼센트로 될수 있기때문이다. 결국 Calc_Gross_Pay 의 출구는 Calc_Medical 의 입구로 된다.

Calc_Medical 은 총액에서 인체보험금을 던 결과를 처리한다. 이 결과는 Calc_Tax 의 결과로 보내 져 정확한 정기적인 세금계산에 리용된다. 그림 19-2 에서는 이 과정을 보여 주고 있다.

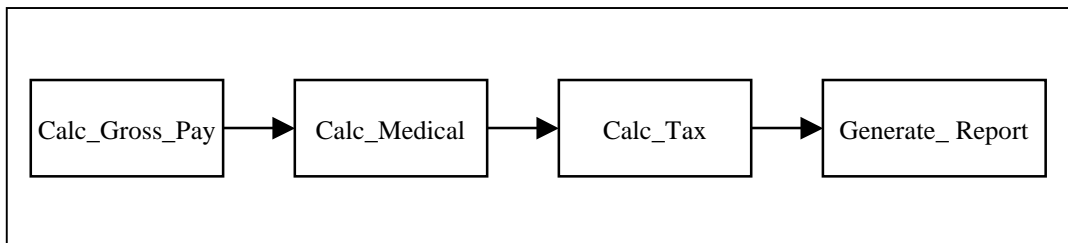



그림 19-2. 하나의 모듈을 두개의 모듈로 가르기

설계는 반복적인 과정으로 진행된다. 즉 어떤 준위로부터 다음준위로 전진할 때 하나의 론리적부분을 몇개의 모듈들로 나누어 가를수 있으며 이것은 결국 프로그람이 더 명백하고 읽기 쉬운것으로 되게 해준다.

실례로 인체보험금에 대한 엄밀한 알고리듬을 개발하는것이 필요한 경우에 지불되는 인체보험금의 형태와 그것의 규정방법을 알아야 한다. 그러므로 추가적설계의 중점은 이 모듈에 있게 되며 이 모듈을 더 많은 모듈로 가를 가능성도 있다.

Calc_Tax 모듈에서는 계산해야 할 국가세금, 지방세금들이 있기때문에 이 모듈에 대하여 더 자세한 연구를 할수 있다.

프로그람작성에서 이러한 과정을 어떻게 일반화할수 있겠는가? 매 모듈은 수행해야 할 기능을 가지며 어떤 의미에서는 하나의 작은 프로그람이라고 볼수 있다. 이러한 작은 프로그람들은 완성된 프로그람을 생성하는데서 하나의 성분으로 된다.

C 언어에서는 모듈을 함수로 고찰하고 있다. Java 와 C++는 함수와 같은 개념을 가지고 있지만 이 언어들은 객체지향언어들이다. 그러므로 여기서 한개의 모듈은 클라스로 될수 있다. 클라스는 함수들을 포함하지만 실지로 클라스는 다음항목에서 론의하는것처럼 더 많은 기능을 가진다.

다음걸음 : 객체지향방법과 설계

만일 C++, Java 혹은 다른 객체지향언어들에서 프로그람을 계획화할 필요가 없다면 이 항목을 보지 않아도 된다. 그러나 프로그람작성을 전부 자세히 계획화하려면 객체지향프로그람작성법에 대하여 잘 알아야 한다. **객체지향프로그람작성법**(Object - Oriented Programming)은 쏘프트웨어개발에서 이룩된 하나의 큰 성과로 된다.

앞에서 고찰한 모듈에 대한 정의는 객체지향적모듈을 취급할 때 더더욱 필요한것이다. 확장가능성, 재리용가능성, 믿음성은 이 항목에서 취급하여야 할 기본목표이다.

확장가능성

모듈은 재리용가능성을 념두에 두면서 설계되므로 프로그람작성자는 모듈에서 그 모듈에 필요한 모든 자료와 연산들을 미리 예견할수는 없다. 그러므로 프로그람작성자는 앞으로의 변경과 확장에 대해서도 생각하면서 모듈을 설계하게 된다. 이러한 방법론을 **확장가능성**(Extensibility)이라고 부른다.

다음과 같은 두개의 원칙들이 확장가능성을 개선하기 위한 열쇠로 된다.

- 설계의 단순성. 단순한 구성방식은 복잡한 구성방식보다 더 쉽게 변경에 적응될수 있다.
- 분산화. 보다 독립적인 모듈들은 변경이 한개의 모듈에서만 진행되도록 한다. 전반적인 구성방식에서의 모듈의 독립성은 프로그람전체를 변경시킬 때 일어나는 련쇄반응을 감소시킬수 있다.

재리용가능성

다른 프로그람들의 구성에 리용될수 있는 쏘프트웨어의 성분들 혹은 모듈들은 **재리용가능성**(Reusability)을 가진다고 볼수 있다.

재리용가능성은 많은 응용프로그람들이 자주 류사한 패턴을 가진다는 견해로부터 제기되는 문제이다. 많은 응용프로그람들은 이미 개발된 쏘프트웨어에 있는 류사하거나 동일한 기능들을 요구하게 된다. 이것은 쏘프트웨어개발의 시간, 로력, 가격을 감소시키는 우점으로 된다.

믿음성

믿음성이 없는 쏘프트웨어는 본질적으로 리용할수 없다. 일부 설계개념들은 믿음성을 높이는데 도움을 준다.

- 호환성을 보장한 설계는 프로그람의 성분들이 다른것들과 쉽게 결합되도록 한다. 이것은 때때로 세계적인 표준규격에 모순될수 있다.

  호환성을 보장하는데서 기본원칙은 표준화된 형식에 맞추면서 동등하게 설계하는것이다. 이 원칙은 작업그룹, 회사 그리고 쏘프트웨어개발집단안에서까지도 지켜야 한다.
  실례로 한 작업그룹은 기계에 의존하는 2진파일이 아니라 문자렬을 포함하는 본문파일로부터만 통보를 추출할수 있다고 하자. 만일 이 작업그룹의 일부가 Windows NT 체계의 프로그람과 UNIX 체계의 또 다른 프로그람을 개발한다면 이 경우에 프로그람의 재리용성은 대단히 큰 문제로 제기된다고 볼수 있다.

- 효과성이 높은 설계는 처리장치의 시간, 차지한 기억공간, 통신에 리용되는 주파수범위와 같은 하드웨어자원들에 대한 요구들을 최소화하게 한다. 만일 프로그람이 효과성에 대한 고려가 없이 설계되였다면 그 프로그람이 작은 체계에서 수행되는 경우에 수행시간이 길거나 많은 자원을 리용하게 될것이다.

  오늘날 프로그람에서 그것의 성능과 측정가능성은 중요한 문제로 제기되고 있다. 효과성을 념두에 두면서 설계한 프로그람은 더 빨리 수행되고 더 적은 자원을 리용할것이다. 설계에 이러한 특징들을 반영한 프로그람은 총체적으로 자원을 거의 리용하지 않으면서도 더 많은 처리를 하게 된다.

- 만일 호환성이 고려되지 않으면 쏘프트웨어는 한가지 형태의 콤퓨터체계에서만 효과적인것으로 된다. 몇가지 훌륭한 호환성을 보장하면 다른 호환성들도 조화시킬수 있다.

하드웨어는 오늘의 환경에서 빠르게 서로 잘 어울리는 상품으로 되여 가고 있다. 임의의 하드웨어환경에서 수행될수 있는 프로그람이 리상적인 프로그람이다. 그러나 아직은 다른 하드웨어환경에서의 수행을 위하여 최소한의 변경이 필요한 프로그람도 허용될수 있다.

호환성은 계속 고찰할 필요가 있는 문제점이다. 프로그람의 호환성을 100%로 보장하면 그 프로그람의 성능은 떨어 질수 있다. 성능을 최소한으로 보장하면서 호환성을 보장하기 위하여 프로그람을 변경시키려면 쏘프트웨어검사에 대한 요구와 IT조직에 대한 더 많은 투자가 필요하게 된다.

- 사용의 편리성은 프로그람을 사용자가 누구나 다 쉽게 그 사용법을 배울수 있고 그것을 문제풀이에 적절히 적용할수 있도록 담보해 준다.

  직관적인 프로그람의 작성은 소리를 내는 일처럼 항상 쉬운것은 아니다. 일부 IT조직들은 대면부설계에 대한 전문지식을 가진 사람들을 받아 들여 리용하고 있다. 만일 그렇게 할수 없으면 다른 사람들이 자기의 설계를 다시 조사하도록 하여야 한다. 다른 사람들은 자기의 설계를 잘 모르기때문에 명백히 설명해 주어야 한다.

- 검증성은 프로그람이 자기의 실패를 판단하고 오유를 추적할수 있게 설계된다는것을 의미한다.

  프로그람수행에서의 실패는 대부분 프로그람들에서 다 나타난다. 그러나 그 실패는 반드시 프로그람적인 흐름에 의해서만 생기는것이 아니다. 이러한 실패는 하드웨어문제, 외부적원천으로부터의 잘못된 정보, 사람에 의한 오유 등에 의해서도 생길수 있다. 그러므로 파일로부터의 읽기나 쓰기와 같이 자주 발생되는 오유들을 프로그람에서 조종할수 있도록 담보해 주어야 한다.

- 한개의 쏘프트웨어체계에 맞게 설계된 완전성은 비법적인 접근과 수정으로부터 프로그람을 보호할수 있게 한다.

안전성과 비밀보장은 오늘날 대단히 큰 문제로 제기되고 있다. 프로그람이 수행되는 하드웨어를 **해커**(Hacker)들로부터 안전하게 보호하는데 주의를 돌려야 한다. 그러자면 프로그람에 안전성보장을 위한 기능들을 가능한껏 도처에 포함시켜야 한다. 이러한 기능들로서는 가입자접근, 하드웨어방화벽과 쏘프트웨어방화벽, C++객체에서의 정보의 국부화 등을 들수 있다. 또한 이러한 기능들을 결합시킬수 있는 여러가지 방법들도 존재한다.

이러한 조종방법들은 개별적인 언어들에 대하여 학습할 때 더 고찰하게 된다. 매개 언어는 자체로 이러한 기능들을 포함하고 있다.

다음항목에서는 여러가지 프로그람작성에 대한 표준적인 방법론들을 고찰한다.

절차적방법

초기의 프로그람작성방법은 절차적방법이다. 절차적인 프로그람작성법에서는 문제를 해결하기 위한 걸음이 무엇인가에 기본중점을 둔다. 이 방법에서는 프로그람의 순서를 창조하는데 함수 혹은 부분프로그람을 리용한다. 프로그람은 여러개의 함수들로 구성되며 이 함수들은 자료를 조종하게 된다.

절차적방법을 리해하기 위하여 그것이 어떻게 작업하는가에 대한 실례를 고찰하자.

여기서 고찰하는 프로그람은 종업원들의 월별 생활비를 관리하고 생활비를 생성하는 프로그람이다. 우선 EMPLOYEE 라고 부르는 자료구조를 창조한다. 이 자료구조는 종업원들의 NAME 과 SALARY 를 포함한다. 여기에 종업원들에 대한 생활비제고마당을 포함시킬수도 있다. 또한 IncresePay()라는 함수를 정의한다. EMPLOYEE 자료구조는 이 함수의 입력으로 사용된다. 그리고 이 함수는 종업원들의 생활비를 증가시키고 그것을 EMPLOYEE 자료구조안에 있는 SALARY 에 기억시킨다.

이 프로그람은 앞으로 확장될것을 요구할수 있기때문에 프로그람이 단순하다고 해도 절차적프로그람작성자들에게 있어서 일정한 문제가 제기될수 있다. 즉 직업명, 전화번호, 장소, 시민증번호와 같은 여러가지 정보를 EMPLOYEE 자료구조에 포함시킬수 있다. EMPLOYEE 자료구조에 대한 변경 혹은 추가는 그 자료구조를 리용하는 함수까지도 변경시키게 한다.

프로그람의 복잡성이 점점 커지는데 따라 그것을 관리하고 확장하는 능력도 역시 커진다. 절차적방법에서는 자료와 자료를 조종하는 함수사이의 내부적련관을 전혀 보장하지 않는다. 또한 함수의 기능을 수정해야 한다면 그것을 다시 쓰지 않고는 쉽게 고칠수 없다. 실례로 생활비를 증가시키는 복잡한 방법을 리용하려는 경우에 이미 존재하는 코드를 재리용하는 유일한 방법은 그 함수를 제거하고 새로운 함수를 추가하는것이다.

문제점은 점점 더 복잡해 질수 있다. 오늘날 세계적으로 절차적프로그람작성법은 사용자들에게 더는 만족을 주지 못하고 있으며 새로운 객체지향적인 방법론이 절실하게 요구되고 있다.

객체지향방법

객체지향언어(Object - Oriented Language)들은 프로그람작성자들에게 자료구조와 자료조종방법(method 혹은 function)들사이에 강력한 련계를 지어 줌으로써 현실세계를 보다 세밀하게 모형화할수 있게 해준다. 더 중요하게 프로그람작성자들은 자료구조와 자료조종함수들에 대하여 더는 고찰하지 않고 그대신 **객체**(Object)를 다룰수 있게 해준다.

객체지향언어들에서는 기본적인 블로크작성을 위한 개념으로서 **클라스**(Class)와 객체를 사용한다. 객체지향프로그람작성법은 현실세계를 받아 들이는 방법론을 주기때문에 현실세계에 대한 실례를 리용하여 설명하는것이 제일 좋다. 많은 사람들은 자동차를 운전할줄은 알지만 자동차가 어떻게 만들어 졌는가에 대해서는 잘 모른다. 왜냐하면 그것이 자동차를 리용하는 관점에서는 필요가 없기때문이다. 이와 같이 현실세계에는 사용법은 알아도 그것

이 어떻게 만들어 졌는가를 알지 못하는 객체들이 많다. 그 리유는 많은 객체들이 훌륭한 사용자대면부를 제공하도록 설계되였기때문에 사용자들은 그 객체들이 어떻게 실현되였는가에 대한 지식이 없이도 그것을 쉽게 다룰수 있기때문이다. 대면부는 실현방법에 의존하지만 그 실현의 복잡성을 사용자들에게는 숨기고 있다. 따라서 실현과정이 변경되여도 대면부만 변하지 않으면 사용자들은 그것에 대하여 알거나 관심할 필요가 전혀 없다.

만일 어떤 사람이 최신형의 승용차를 구입하였다면 그 사람은 승용차의 기관이 어떻게 설계되였는가에는 관심하지 않는다. 또한 새 형태의 승용차가 이전의것보다 성능이 좋다고 하여도 사용자는 그 승용차를 운전하는 방법에만 관심을 돌린다. 결국 사용자의 기술은 대면부가 변하지 않았으므로 달라 질것이 없게 된다. 만일 승용차제작자들이 바닥에 있던 제동발판대신에 계기판옆에 손제동기를 달려고 한다면 승용차구입자는 새로운 대면부에 맞게 운전기술을 다시 익혀야 한다.

객체지향언어에 대한 개념을 가지는데서 중요한것이 바로 이러한 원리적인 고찰방법을 가지는것이다. 모든 실현의 내용은 훌륭한 대면부뒤에 숨겨 져야 한다. 그러면 사용자들은 그것의 실현내용을 전혀 몰라도 대면부만 알면 된다.

승용차라는 술어만으로 사용자들은 거의 모든것을 리해한다고 볼수 있다. 왜냐하면 눈에 보이지 않는것은 알 필요가 없으며 그것을 몰라도 사용자들은 생활에서 승용차를 능숙하게 리용할수 있기때문이다. 즉 모든 승용차들은 려행에 리용되며 기관, 좌석, 조향장치와 같은 공통적인 속성들을 가지고 있다. 또한 모든 사용자들은 열쇠로 승용차에 발동을 걸거나 제동하며 가속, 감속, 왼쪽 회전, 오른쪽 회전 등과 같은 공통적인 운전기술을 가지고 있다. 본질적으로 사람들은 승용차라는 개념만으로 모든 승용차들의 외형 혹은 구조와 같은 정적속성과 움직임과 관련되는 행동을 포착할수 있다. 바로 이러한 개념이 클라스라고 알려 져 있다. 물리적인 승용차가 바로 객체이다. 즉 승용차객체는 승용차클라스의 한가지 구체례이다. 클라스와 객체사이의 관계를 구체화관계라고 부른다. 승용차객체는 승용차클라스로부터 일반화된다고 말하고 승용차클라스는 모든 승용차객체들의 일반화라고 말한다.

만일 어떤 사람이 책상의 속도가 5 초동안에 0 부터 60 까지 가속되였다고 말한다면 사람들은 아마 그 사람이 미쳤다고 말할것이다. 그러나 승용차의 속도가 5 초동안에 0 부터 60 까지 가속되였다고 말한다면 있을수 있는 일이라고 생각할것이다. 그 리유는 클라스의 이름이 속성모임뿐아니라 그것의 작용에 대한 감각까지 주기때문이다. 이렇게 자료와 작용사이의 관계는 객체지향적방법에서 열쇠로 된다.

교갑화

절차적언어에서 함수에 대한 자료의 종속성은 쉽게 찾아 볼수 있다. 자료의 종속성을 찾아 보려면 함수에 입구되는 자료, 함수에 귀환되는 자료들을 고찰하면 된다. 변수들도 함수안에서 리용될수 있다. 만일 변수들이 함수가 실행될 때마다 함수안에서 창조, 리용, 파괴된다면 그러한 변수들은 국부변수로 서술된다. 이 관계를 그림 19-3 에서 보여 주고 있다.

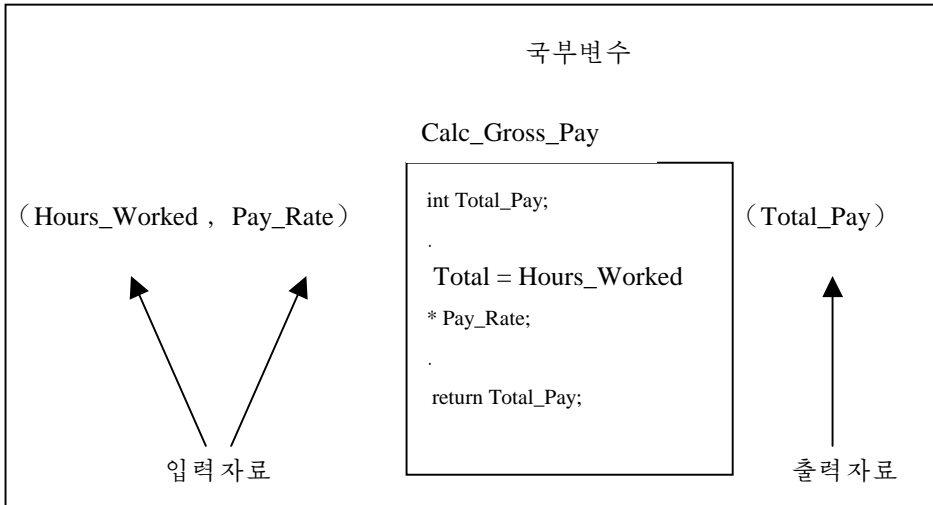



그림 19-3. 국부변수

절차적언어에서 한가지 문제점은 토막자료에 대한 기능적종속성을 찾는 능력이다. 자료에 대한 기능적종속성을 찾는다는것은 그 자료를 리용하거나 혹은 그 자료에 종속되는 모든 기능들을 찾아야 한다는것을 의미한다. 이를 위해서는 이러한 기능들을 찾으면서 코드전체를 검사하여야 한다. 만일 자료가 어떻게 변경되고 어떻게 리용되는가를 알아야 할 필요가 있다면 이 과정은 대단히 크고 오유를 범하기 쉬운 과정으로 된다. 프로그람이 커질수록 이러한 문제점은 더 커진다.

객체지향언어들에서는 이러한 문제점이 해결되였다. 여기서는 기능에 대한 자료의 종속성과 자료에 대한 기능의 종속성을 둘 다 고찰할수 있다. 객체들은 구체레들과 함께 완전히 결합되여 있다. 객체들이 알수 있는 모든것(자료)과 객체들이 할수 있는 모든것(방법)들이 순수한 덩어리로 결합되여 있다. 이러한 결합을 **교갑화**(Encapsulation)라고 부른다.

생활비를 고찰하는 프로그람에서 EMPLOYEE 클라스의 사용자는 종업원생활비가 클라스안에서 어떻게 기억되는가를 알 필요가 없다. 모든 사용자들이 관심하는 부분은 Total_Pay 가 EMPLOYEE 형태의 객체를 탐색할수 있다는것이다. 이 과제는 GetPlay()라는 방법에 의하여 호출된다. 사용자는 EMPLOYEE의 성원함수 GriveARaise()를 호출함으로써 종업원의 생활비를 높여 줄수 있다.

방법들인 GetPay()와 GiveARaise()는 EMPLOYEE 클라스에 포함되며 EMPLOYEE 의 전역적인 대면부로 된다. 대면부는 EMPLOYEE와 그 사용자들이 호상 작용하게 해준다. 즉 사용자들에게 클라스가 무엇을 할수 있는가를 알려 준다. 이에 대하여 그림 19-4 에서 보여 주고 있다.

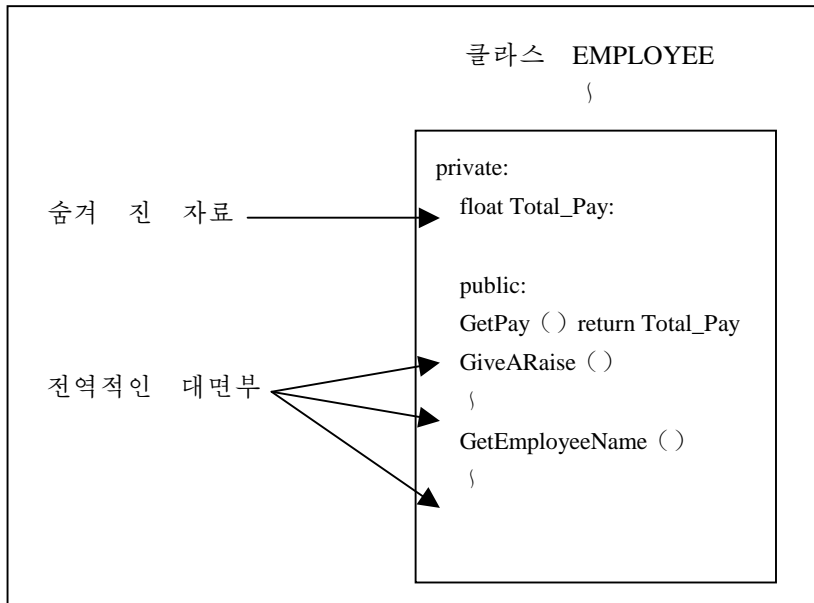



그림 19-4. 교갑화

우의 실례에서 사용자는 EMPLOYEE의 전역적인 대면부를 통하여 다음과 같은 질문을 할수 있다.

- 어떤 종업원의 이름을 찾으시오(GetEmployeeName()).
- 어떤 종업원의 생활비를 높여 주시오(GiveARise()).
- 종업원에게 지불한 총액을 계산하시오(GetPay()).

사용자는 클라스 EMPLOYEE 안에 있는 변수 Total_Pay 를 직접 변경시키거나 볼수조차 없으며 다만 클라스 EMPLOYEE 의 방법을 통해서만 접근할수 있다. Total_pay 에 대하여 수행되는 모든 작용은 클라스 EMPLOYEE 의 방법들인 GetPay()와 GiveARise()들을 통하여서만 진행할수 있다.

Total_Pay 는 성원변수이다. 성원변수는 클라스의 부분으로 된다. 만일 후에 Total_Pay 가 자료기지로부터 제거된다고 하여도 아무런 영향을 주지 않는다. Total_Pay 의 내부적실현은 사용자들에게 보여 지지 않는다. 이것을 **정보은페**(Information Hiding) 혹은 **자료은페**(Data Hiding)라고 부른다. 이것이 교갑화에 대한 기본견해이다. 사용자들이 전역적인 대면부를 사용하는 한 국부적인 자료에 대한 접근은 그 자료들이 어떻게 기억되여 있는가에는 무관계하게 방법들을 통하여서만 할수 있도록 담보된다.

계 승 성

계승성은 객체지향언어들사이의 보다 중요한 관계의 하나로 되고 있다. 계승성은 클라스들사이의 **일종관계**(a-kind-of 관계)로 고찰된다. 실례로 PordTaurus 는 승용차의 한가지 종류이다. 또한 개는 짐승의 일종이며 달마띠야개는 개의 일종인것처럼 관계는 본질적으로 계층화될수 있다. 계층화의 1 차적목적은 다음과 같은 두가지이다. 하나는 두개의 클라스들사이에 공통적인 표현을 허용한다는것(일반화)이고 다른 하나는 한개 클라스가 다른 클라스의 특수한 형태로 된다는것(특수화)을 지적하는데 리용될수 있다는것이다.

보통 프로그람의 첫 방안에서 설계자들은 일반화를 하려고 한다. 설계자들은 두개이상의 클라스들이 자료, 행동, 공통대면부와 같은 공통적인것들을 가지도록 한다. 이 정보들은 표준적으로 클라스들이 계승할수 있는 보다 일반적인 클라스에 집합된다.

실례로 만일 클라스 Dalmatian 와 Dachshund 가 있을 때 그것들사이에는 그림 19-5 에서 보여 준것처럼 몇가지 공통점들도 있고 차이점들도 있다.

Dalmatian

2 개의 귀
2 개의 눈
1 개의 코
4 개의 다리
검은 반점들이 있는 흰 털
식욕이 높다.
말들을 좋아한다.

Dachshund

그림 19-5. 공통점과 차이점

이것들은 차이가 심한 두개의 개품종들이지만 2 개의 귀, 2 개의 눈, 1 개의 코, 4 개의 다리와 같은 공통적인 개의 속성들을 가지고 있다. 이러한 속성들을 Dog 라는 다른 클라스에 넣고 Dalmantion 과 Dachshund 클라스들이 그 속성들을 계승하도록 할수 있다.

한편 일반화된 클라스를 고찰하면서 프로그람은 수정을 요구한다는데로부터 특수화(전문화)에 대한 개념이 더 심화되게 되였다. 설계가 전개되는데 따라 프로그람작성자들은 보다 전문적인 취급을 요구하는 클라스들을 실현하게 된다. 바로 이러한 특수경우들을 실현하기 위한것이 계승성이다.

클라스 Dog 는 사용자의 견지에서 보면 너무도 일반적이라고 말할수 있다. 왜냐하면 사용자들은 보통 군견, 사냥개, 관상용개와 같은 특수한 품종의 개에 대한 정보

를 요구하기때문이다. 특수화는 일반화된 클라스로부터 시작하고 그것을 새 클라스의 한 부분으로 계승하는 방법으로 진행할수 있다. 새 클라스는 특정한 요구에 맞게 전문화될수 있다.

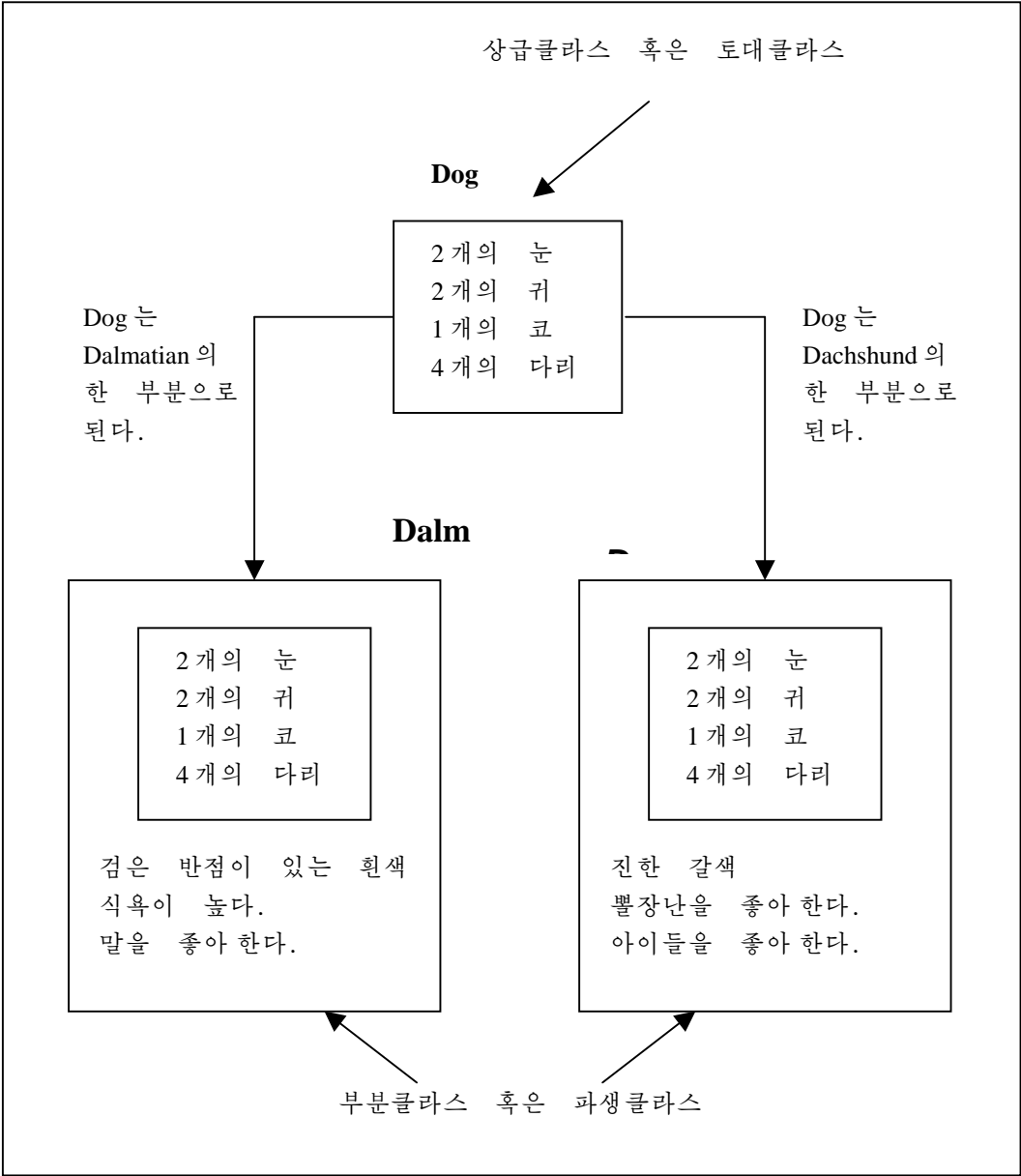



그림 19-6. 자료의 계승성

계승성은 표준적인 성분들을 프로그람의 특수령역에서 받아 들일수 있기때문에 재리용성의 수단으로 고찰할수 있다.

계승성을 리해하려면 몇가지 술어들을 리해할 필요가 있다. 만일 어떤 클라스가 다른 클라스로부터 계승되였다면 그것을 **부분클라스**(Subclass) 혹은 **파생클라스**(Derived Class)라고 부르며 어떤 클라스를 다른 클라스가 계승하였다면 그 클라스를 **상급클라스**(Superclass) 혹은 **토대클라스**(Base Class)라고 부른다.

그림 19-6에서는 자료의 계승에 대한 도식적견해를 보여 주고 있다.

클라스 Dog는 토대클라스이고 클라스 Dalmatian과 Dachshund는 클라스 Dog를 계승하고 있다. 여기서 개를 묘사하는 공통적인 속성들의 모임이 클라스 Dog에서 한번만 서술된다. 주어 진 임의의 클라스는 모두 개와 관련된것이지만 일반적인 개가 아니며 토대클라스 Dog로부터 파생될수 있고 특수한 속성들을 추가하여 새로운 종류의 개를 창조할수 있는 클라스이다.

계승성은 코드를 재리용하기 위한 좋은 방법으로 된다. 사용자는 모든 론리적부분들을 재구축하지 않고 새로운 클라스를 창조함으로써 코드를 재리용할수 있으며 이미 만들어 져 오유처리까지 다 되여 있는 클라스들을 통합할수 있다. 즉 이 방법은 원천코드를 제거하고 추가하는 방법보다는 확실히 더 좋은 방법으로 된다.

다 형 성

다형성(Polymophysm)은 객체지향언어에서 본질적인 기능의 하나이다. 다형성에 의하여 개발자는 클라스를 초기에 창조할 때뿐아니라 새로운 기능이 필요할 때에도 유연하게 발전시킬수 있다.

개들을 고찰하는 클라스에서 개발자들은 클라스의 사용자들이 요구하는 일부 개품종에 대한 규정을 놓칠수 있다.

클라스들은 실제적인 개품종들을 콤파일할 때가 아니라 실행할 때에 련결시키는 방법(얽힘을 해소하는 방법)으로 설계된다. 이것은 토대클라스 Dog가 어떤 특수한 개품종을 념두에 두고 씌여 지지 않는다는것을 의미한다. 토대클라스는 모든 개들의 일반적인 측면만을 고찰한다. 토대클라스의 초기작성자는 어떠한 개품종이 필요한가에 대해서는 알 필요가 없기때문에 주의를 돌리지 않는다. 얽힘을 해소한다는 것은 후날에 특수한 개품종이 결정될 때까지 "가상적인" 개를 고찰한다는것을 의미한다.

실례로 매개 개품종들은 자기의 고유한 개 짖는 소리를 가진다. 토대클라스 Dog에서 임의의 개품종과 개 짖는 소리를 조종한다고 하자. 그러면 토대클라스 Dog의 성원함수 bark()를 호출할 때 그 함수가 어떤 개품종에 해당되는가에 대해서는 알 필요가 없다. 이것은 마치도 사람들이 일상 생활에서 개의 품종에는 무관계하게 개 짖는 소리를 내게 하는것처럼 토대클라

스 Dog의 성원함수 bark()만을 호출하는것과 같다.

토대클라스 Dog의 성원함수 back()를 호출하면 해당한 개품종의 bark()를 호출하는것으로 된다. 그러나 토대클라스 Dog에서 bark()가 가상화된후에라도 사용자는 토대클라스를 계승하여 새로운 파생클라스를 창조할수 있다. 새롭게 창조되는 파생클라스는 특수한 개품종인 Jack Russell Terrier라는 관상용개일수 있다. 파생클라스개발자는 어떤 개품종이 설계되였으며 그 품종의 개들이 어떻게 bark()소리를 내는가를 알고 있다. 그러므로 토대클라스 Dog의 가상적인 bark()를 파생클라스의 특수한 bark()로 교체하면 정확한 bark()함수를 리용하게 된다. 이렇게 토대클라스를 수정함이 없이 계승의 방법으로 새로운 기능을 추가하여 프로그람을 확장할수 있다.

이 실례가 다형성이 없이 실현되였다면 토대클라스개발자들은 필요한 개품종들을 다 고려하여야 한다. 토대클라스에서 Dalmatian, Jack Russell Terrier, Labrador Retriver들을 조종한다고 하자. bark()를 호출하면 그것이 어떤 개품종에 대한것인가를 고려해 보아야 한다. 그러면 클라스 Dog는 정확한 bark()가 아니라 틀린 bark()를 줄수 있다. 또한 토대클라스 Dog에서 규정되지 않은 삽살개에 대한 bark()를 호출하면 그 클라스는 삽살개에 대한 정보를 포함하도록 재편집되고 재콤파일, 재검사되여야 한다.

다형성에 대한 개념은 보통 C++에서 서술되므로 C++프로그람작성언어에 대하여 한번 더 학습하면 명백해 질것이다.

객체지향언어에서의 설계방법

체계가 무엇을 해야 하는가에 대하여 잘 알려면 해당한 조건을 정확히 주는것이 필요하다. 실례로 생활비지불체계에서 오늘이 월의 마지막날이라면 이 체계가 무엇을 해야 하는가 하는 식으로 고찰하여야 한다. 왜냐하면 체계(프로그람)가 규정된 사건이 발생할때에만 해당한 동작을 수행하기때문이다.

이렇게 가능한 경우를 고찰하는 방법은 프로그람에서 몇개의 기본클라스들이 창조되여야 하는가를 구별하게 하는 열쇠로 된다. 프로그람을 설계할 때 어떤 클라스들이 있어야 하며 그것들이 서로 어떻게 관계되는가를 밝힌 다음 매개 클라스를 작성하여야 한다.

- 클라스의 이름. 클라스의 이름은 해석적으로 달아 주어 후에 그것을 참조할 때 그 클라스가 무엇을 하는가를 알수 있게 하여야 한다.
- 클라스가 무엇을 해야 하는가 하는 임무. GetPay()나 GiveARaise()와 같이 성원함수들의 이름을 써야 한다.
- 다른 클라스들과 서로 어떻게 작용하는가 하는 협력관계

사용자는 모든 클라스들을 찾아 낼수도 없고 그것들의 임무와 호상관계들을 전부 결정할수도 없다. 그것은 설계과정이 진척되는데 따라 점차적으로 결정된다.

객체의 설계는 프로그람을 작성하는 시점에서만 할수 있는것이 아니라 여러 단계를 거쳐 진행된다. 오히려 객체가 무엇을 하고 그것이 어떻게 보여야 하는가에 대한 리해보다도 즉시 프로그람을 완성할수 있게 노력하여야 한다. 이러한 리해의 기본 단계들을 아래에서 서술하고 있다.

이 단계들은 Bruce Eckel의 문헌 "Thinking in C++"에 개괄되여 있다.

- 프로그람의 초기해석단계에서 객체는 외부적인 요인들과 제한성들을 검사함으로써 발견될수 있다. 또한 프로그람안에서 성분들이 중복되는가를 찾아 보아야 하며 보다 작은 론리적인 단위 혹은 모듈들이 많은 객체들을 포함하고 있는가를 확인해야 한다.
- 객체를 작성하는 단계에서 새로운 객체에 대한 요구가 제기될수 있다. 객체를 내부적으로 함수화하면 새로운 클라스를 요구할수 있다.
- 클라스들을 함께 결합시켜 프로그람을 구축할 때 객체에 대한 보다 많은 요구를 발견할수 있다. 프로그람에서 객체들사이의 통신과 내부적영향도 새로운 클라스를 요구하거나 이미 존재하는 클라스의 변경을 요구할수 있다.
- 완성된 프로그람에 새로운 기능을 추가하면 앞선 설계에 대한 부족점들이 드러난다. 이미 존재하는 프로그람을 확장하면 새로운 클라스들의 추가를 요구할수 있다.
- 클라스설계의 완전성을 보장하는데서 객체의 재리용이 중요하다. 만일 클라스가 완전히 새로운 상태에서 재리용된다면 일부 결함들이 나타날수 있다.

객체설계는 엄밀하게 체계화된 학문이 아니라 끊임없이 발전하는 학문이다. 우에서 보여 준것처럼 객체설계는 클라스가 존재하는한 계속 발전해 나간다.

제20장. 개 발

자기자신을 위한 프로그람 혹은 어떤 큰 개발과제를 위한 프로그람을 작성할 때 개발과정과 개발도구들을 잘 아는것은 아주 중요하다.

프로그람을 쓰는것 그자체는 개발과정에서 간단한 부분일뿐이며 쏘프트웨어의 질과 완전성을 반드시 고려하는것이 중요하다. 그러므로 순환적인 개발생명주기안에서 개발작업을 함으로써 쏘프트웨어의 오유들을 밝혀 내고 더 좋은 제품으로 만들어야 한다.

프로그람작성자들은 원천코드쏘프트웨어를 관리하는 도구들을 리용하여 개발과정에 제기되는 여러가지 갱신판들과 함께 쏘프트웨어에 취해 진 수정내용들을 관리하고 기록해 두게 된다. 이러한 도구들은 특히 여러 프로그람작성자들이 모여 큰 개발과제를 수행할 때에 유용한것이다. 또한 프로그람작성자들은 개발과정에 진행된 변경내용을 잃어 버리지 않도록 하기 위하여 작업파일을 덧쓰기하여 보관한다.

이 장에서는 여러가지 개발생명주기들을 고찰하고 UNIX 체계에서 대표적인 원천코드조종도구인 SCCS를 고찰한다.

개발생명주기

개발생명주기에 대한 리해는 여러 개발자들이 모여 큰 과제에 대한 프로그람을 작성할 때에 더 필요한것으로 된다. 개발생명주기에서는 프로그람작성자들, 쏘프트웨어검사자들, 프로젝트관리자들이 서로 조화롭게 어울려 작업하게 된다. 개발생명주기의 매개 부분에는 일정한 쏘프트웨어적인 기대값들이 설정되는데 기대값들은 문법적 혹은 론리적 오유들이 밝혀 지고 다시 수정되도록 한다.

쏘프트웨어프로젝트는 세개의 기본적인 분석단계 즉 해석단계, 개발단계, 검사단계로 나눌수 있다.

해석단계

매개 프로그람에는 그것을 리용하는 사용자들의 고유한 집단이 있다. 사업상 변화는 새로운 프로그람의 창조 혹은 이미 존재하는 프로그람의 확장을 요구하게 하며 여기에 특별한 관심을 돌리게 한다. 결국 이러한 요구는 개발집단의 임무로 나선다. 따라서 코드작성자들과 창안자들은 이 사업을 위한 총괄적인 해석을 하고 프로그람에 필요한 변경내용을 개괄하게 된다. 설계자들은 이러한 변화를 모형화하여 설계문서를 창조하거나 개선시킨다.

개발단계

프로그람작성자들은 설계문건들에 반영된 요구조건에 맞게 원천코드를 수정하거나 새롭게 창조한다.

프로그람작성자들은 편집기를 리용하여 원천파일을 창조하거나 편집하게 된다. 원천파일은 프로그람작성언어로 씌여 진 읽기가능한 본문지령들로 된 원천코드를 포함한다.

원천코드가 완성된후에 개발자는 프로그람을 콤파일한다. 콤파일러는 프로그람작성언어의 규칙에 맞지 않게 본문지령들을 써넣은 경우에 발생하는 문법적오유들을 검사한다. 만일 문법적오유들이 나타나면 콤파일러는 원천코드를 다시 수정하고 그것을 재콤파일하여야 한다. 이 과정은 콤파일이 성과적으로 완성되고 실행가능한 프로그람이 얻어질 때까지 계속된다.

검사단계

검사단계에서는 제일 먼저 **단위별 검사**(Unit Test)가 진행되며 여기서 프로그람작성자들은 새롭게 창조되거나 개선된 프로그람작성의 론리적부분들을 검사하게 된다. 그리고 설계단계에서 제기된 요구조건에 맞게 프로그람이 수행되고 결과가 나오는가에 대하여 검사하게 된다.

프로그람작성자에게 통과되면 원천코드는 **통합검사**(Integration Test)에 넘겨 진다. 여기서는 프로그람작성자가 자기의 원천코드를 다른 사람의 원천코드와 통합시켜 검사한다. 이 단계는 하나의 프로그람에서 작성하는 프로그람작성자가 여러명인 경우에 특히 더 중요한 단계이다.

통합검사단계가 성과적으로 완성되면 원천코드는 **체계검사**(System Test)로 넘겨진다. 많은 프로그람작성과제들은 조직적인 검사단체들을 가지고 있으며 이 조직에 의하여 프로그람전체가 사용자들과 설계의 요구조건에 맞는가를 검증 받게 된다. 만일 어떤 단계에서 론리적인 오유가 나타나면 그것은 프로그람작성자들의 책임으로 된다. 그러나 오유가 설계흐름의 잘못이라면 개발과정은 설계단계로부터 다시 진행되여야 한다. 진퇴량난이 발견된 후에 순환은 오유가 나타난 그 시점으로부터 다시 시작되여야 한다.

최종적인 검사단계를 **승인검사**(Acceptance Test)라고 부른다. 프로그람은 생산환경 혹은 생산환경을 모방한 개별적콤퓨터체계에서 검사된다.

승인검사가 끝나면 프로그람은 생산환경으로 넘겨 지며 사용자집단에서 사용할수 있게 된다.

다른 프로그람작성단체들에서는 이러한 과정에 대한 여러가지 방안들을 가지고 있으며 여기서 다른 용어를 사용하기도 하고 보충적인 단계를 추가하기도 하며 다른 단계들

을 통합하기도 한다. 어쨌든 모든 조직들은 론리적부분과 설계부분에서의 오유를 느끼게 되며 그렇게 되면 이전단계로 되돌아 가서 반복작업을 하게 된다.

개발단계에서 발견된 오유들은 분석단계에서 발견된 오유보다 더 많은 시간과 비용을 소비하게 된다. 더우기 검사단계에서 발견된 오유는 훨씬 더 많은 비용을 요구하며 생산단계에서 오유가 발생하면 그 수정에 최대의 비용을 소비하게 된다.

이처럼 문제의 분석과 설계는 대단히 중요하다는것을 알수 있다. 분석과 설계에서 명확하지 않은것들이 뒤공정들에서 무시될것이라고 가정하면 실천단계에서 좋은 결과를 얻을수 없다. 또한 개발생명주기안에서 자세히 고찰하지 못한 문제들은 막대한 노력과 비용, 시간을 랑비하게 될것이다.

SCCS 원천코드조종체계

많은 프로그람들은 한개이상의 파일 때로는 많은 파일들로 구성된다. 하나의 프로그람에 대한 원천코드는 여러개의 원천파일안에 들어 갈수 있다. 이러한 파일안에 존재하는 원천코드로서 프로그람은 그 프로그람자체를 유지하게 된다. 따라서 이러한 매개 파일의 여러가지 방안들을 관리하는것은 힘든 일로 된다. 원천파일과 문서파일들은 오유가 수정될 때마다 자주 변하며 이에 따라 프로그람도 발전하고 쏘프트웨어의 새로운 방안이 생성되게 된다. 사용자들은 프로그람작성자들이 새로운 방안을 개발하는 동안에는 낡은 방안을 사용하게 된다. 그러므로 이러한 프로그람의 새로운 방안들을 추적하면서 이미 진행한 변경내용을 잃어 버리기가 쉽다.

원천파일들은 보통 하나의 등록부안에 있으며 개발자들이 접근할수 있다. 다음과 같은 경우를 고찰하자.

개발자 A 는 원천파일의 변화를 요구한다. 개발자 A 는 원천파일이 기억되여 있는 전역적인 등록부로부터 그 원천파일을 자기의 홈등록부에 복사한다. 그리고 A 는 이 파일에서 원천코드에 대한 변경을 진행한다.

한편 개발자 B 도 꼭 같은 원천파일의 변화를 요구한다. 개발자 B 도 역시 전역적인 등록부로부터 자기의 홈등록부에 원천파일을 복사하며 이때 개발자 B 는 A 가 이 파일을 복사하고 현재 그 파일에 대한 변경을 진행하고 있다는것을 전혀 알지 못한다. 개발자 B 도 변경을 진행한다.

개발자 A 는 원천파일에서 원천코드의 변경을 완성하였다. 개발자 A 는 원천코드를 성과적으로 콤파일하고 코드의 단위별검사를 진행한다. 그리고 개발자 A 는 원천파일이 기억되여 있던 전역적인 등록부에 원천파일을 귀환한다. 개발자 A 는 통합검사를 성과적으로 수행한다.

개발자 B 는 원천파일에서 원천코드의 변경을 완성하였다. 개발자 B 는 원천코드를 성과적으로 콤파일하고 코드의 단위별검사를 진행하며 원천파일이 기억되여 있던 전역적인 등록부에서 개발자 A 가 복사한 원천파일에 덧쓰기한다. 개발자 B 와 개발

자 A 는 이러한 일이 진행된데 대하여 서로 알지 못한다. 개발자 B 는 통합검사를 진행한다.

쏘프트웨어는 체계검사에로 넘어 간다. 외부적인 조직은 검사를 시작하며 거기에 개발자 A 의 변경이 없다고 지적할것이다. 개발자 A 는 이 통보를 받고 자기의 작업을 다시 반복해야 할것이다.

이러한 현상을 피하자면 어떻게 하여야 하겠는가? 이 문제를 해결하기 위하여 UNIX 는 원천파일의 변경을 추적하고 관리하는 봉사프로그람 SCCS(Source Code Control System)를 제공한다.

SCCS 는 조직들과 개별적사람들이 자기의 파일들을 조종할수 있게 해 준다. SCCS 는 파일의 갱신을 허용하는 사람들과 허용하지 않는 사람들을 판단하게 된다. SCCS 는 파일의 갱신이 일어 날 때마다 변경의 리유를 고려하여 설명 혹은 표식도 붙인다. 또한 SCCS 는 변경이 진행되는데 따르는 원천파일의 이전방안을 기억하고 있으므로 그것을 재생할수도 있다. SCCS 는 새로운 방안에 대한 전면복사가 아니라 변경내용만을 기억시키도록 함으로써 SCCS 의 파일의 크기를 적당하게 유지하도록 한다. 파일에 대한 변화의 리력도 요구될수 있으며 이때 SCCS 는 리력을 방안, 날자, 시간, 저자, 설명문들로 출구한다.

SCCS 는 개발자들에게 읽기전용복사도 보장해 준다. 또한 SCCS 는 편집가능한 파일을 취할수 있게 해주며 편집한 파일을 열쇠를 걸어 덧쓰기하는것으로부터 방지하도록 해 준다.

SCCS 의 개정판본번호붙이기

SCCS 파일이 편집되고 귀환되면 그 변화는 기록된다. 이때 그 변화는 **증분**(Delta)으로 고찰된다. 쏘프트웨어에 대한 변화는 쏘프트웨어에 대한 **개정**(Revision)으로 고찰한다.

매 증분은 두개 혹은 네개 성분으로 이루어 진 판본번호를 가진다. 첫 두개의 수는 항상 있어야 하는것으로서 발행번호와 준위번호이다. SCCS 파일이 초기에 창조되였을 때 그것들의 기정값은 Version 1.1 혹은 Delta 1.1 이다. 련속적으로 개정판이 나오면 SCCS 는 자동적으로 1.2, 1.3 등과 같이 번호를 붙인다. 사용자들도 판본번호를 조종할수 있으며 준위번호는 뛰여 넘을수도 있고 발행번호는 변경시킬수도 있다. 원칙적으로 발행번호는 쏘프트웨어에 대한 중요한 변경이 진행되였을 때에만 Version 1, Version 2 로 변한다.

파일의 방안들은 순차적인 형태로 되여 있으며 매개 증분이나 개정들은 이전의 증분 혹은 개정내용들을 포함하여야 한다. 그림 20-1 에 있는 마지막칸은 쏘프트웨어에 대한 주되는 개정을 반영하기 위하여 발행번호에 대한 변화를 보여 준다.

S

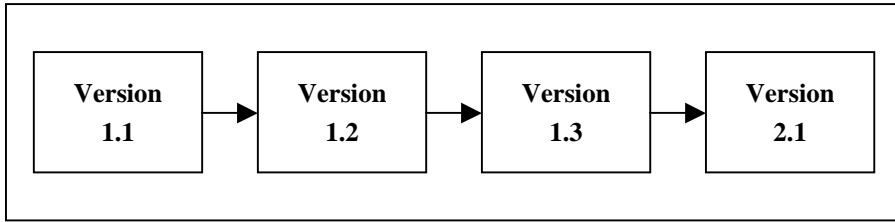



그림 20−1. SCCS 쏘프트웨어의 개정판의 순차적인 형태

그러나 중간방안이 필요한 경우도 있다. 실례로 쏘프트웨어의 판본 2.1 은 개발중에 있고 그 이전판본인 1.3 은 현재 생산에 도입되여 있다고 하자. 앞으로 판본 1.3 의 도입중에 오유가 발견되고 돌발적인 사태가 나타날수 있다. 그러나 판본 2.1 이 아직 생산에 도입될만큼 완성되지 못하였다. 따라서 이때에는 제작파일의 중간방안이 필요하다. 이것은 그림 20−2 에서 보여 준것처럼 가지를 쳐나가게 된다.

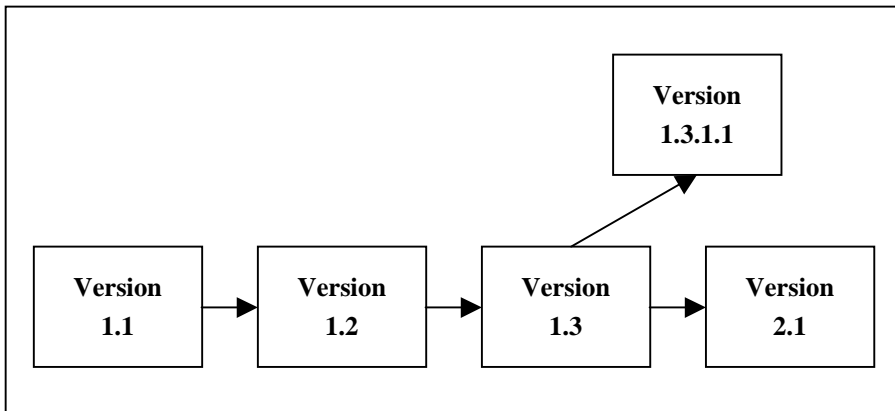



그림 20−2. 가지친 중간판본

판본 1.3.1.1 에서 시작되는 가지에서 만들어 진 개정판들은 판본 2.1 과 그 이후의 판본들에도 포함되지 않게 된다. 가지친 판본에 만들어 진 모든 개정판들은 초기의 판본흐름에 영향을 주지 않는다. 만일 판본 2.1 에서 오유가 나타났다면 그 오유에 대한 수정은 새로운 판본 2.2 에서 진행되게 된다.

SCCS 지령

SCCS 는 사용자가 쏘프트웨어판본들을 관리할수 있게 하는 많은 지령들을 제공한다. 모든 SCCS 파일들은 "s."으로 시작하며 그뒤에 SCCS 의 파일이름이 놓인다. SCCS 파일은 맨 처음에 만들어 질 때 "s."라는 파일을 창조하며 이 파일에는 쏘프트웨어의 변경, 그 변경에 대한 설명, 날자, 시간과 쏘프트웨어를 변경시킨 개발자의 ID 그리고 개정정보가 포함된다.

admin

지령 admin 은 가장 중요한 SCCS 봉사프로그람들중의 하나이다. 이 지령은 SCCS 파일들을 창조하고 그 쏘프트웨어에 대한 개정가능성을 조종하며 개정에 대한 요구조건들을 변경시키는데 리용된다.

모든 SCCS 파일들은 어떤 임의의 동작을 수행하기전에 반드시 admin 지령을 써서 빈 파일 혹은 어떤 파일의 내용으로 초기화되여야 한다.

SCCS 파일을 빈 초기의 개정판으로 창조하려면 다음과 같은 지령을 주어야 한다.

```
$ admin -n s.payroll.txt
```

여기서 "s."은 SCCS 파일의 앞붙이이고 payroll.txt 는 SCCS 가 취급할 파일의 이름이다.

어떤 다른 파일의 초기내용을 가진 SCCS 파일을 창조하려면 다음과 같은 지령을 리용한다.

```
$ admin -i payroll.txt s.payroll.txt
```

여기서 SCCS 파일 s.payroll.txt 는 payroll.txt 의 내용으로 초기화된다. SCCS 파일들을 더 간단히 창조하는 방법은 다음과 같다.

```
$ sccs create payroll.txt
```

우와 같이 하면 파일 s.payroll.txt 가 이 지령이 수행된 등록부에 자동적으로 창조된다.

get

지령 get 는 이미 존재하는 SCCS 파일로부터 복사파일을 만들도록 한다. 지령 get 를 리용하면 임의의 판본들은 자기의 판본번호 혹은 날자에 의하여 검색될수 있다.

지령 get 가 아무런 선택항목도 없이 사용되면 SCCS 는 읽기허락만을 가진 제일 마지막 방안의 파일을 생성한다. 왜냐하면 지령 get 가 임의로 변경을 진행하거나 새롭게 개정하지 못하도록 담보해 주기때문이다.

```
$ get s.payroll.txt
  1.1
  4 lines
  No id keywords
```

이 지령이 수행되면 세개 행들이 출구된다. 첫번째 행의 1.1 은 현재 쏘프트웨어의 개정번호이고 두번째 행은 파일안에 있는 행의 개수이며 세번째 행은 얼마나 많은 실마리어들이 확장되였는가를 보여 준다. 이와 같은 읽기전용파일은 이 지령이 수행된 등록부에 만들어 진다.

만일 사용자가 파일을 변경시켜 쏘프트웨어의 새로운 판본을 만들고 싶다면 다음과

같은 지령을 리용한다.

$ **get -e s.payroll.txt**
1.1
new delta 1.2
4 lines

첫번째 행은 현재 개정번호이고 두번째 행은 새 개정번호이며 세번째 행은 파일에서 행의 개수이다. 선택항목 -e 는 파일 payroll.txt 의 가장 최근의 개정판을 편집하여 출력하도록 한다. 우의 실례에서 지령 get 는 쓰기가능한 복사파일을 생성하고 p.payroll.txt 라는 자물쇠를 건 파일을 창조한다. 이 자물쇠는 증분 혹은 지령 unget 를 리용하여 이 파일이 거꾸로 검사될 때까지 다른 사용자들이 편집할 목적으로 같은 판본을 검색하지 못하게 한다. 읽기전용의 복사파일들은 아직 생성되지 않는다.

선택항목 -e 는 동일한 파일에 대한 여러가지 판본들이 동시에 편집될수 있게 한다. 그러나 두개의 판본이 동일한 개정경로에는 있을수 없다. 실례로 만일 초기의 개정경로로부터 하나의 개정이 진행된다면 동시에 진행되는 또 다른 개정은 가지친것으로만 될수 있다.

지령 get 에 의하여 파일이 생성될 때 거기에 붙여 지는 개정번호는 가장 최근의 개정번호이다. 선택항목 -p 와 -c 들을 리용하면 다른 개정번호들이 생성되게 할수 있다. 이 두개의 선택항목들은 선택항목 -e 와 결합될수 있다.

SCCS 는 선택항목 -r 를 리용하여 쏘프트웨어에 있는 개정판본들중에서 제일 높은 번호의 개정판본을 검색할수 있다.

$ **get -r2 s.payroll.txt**
2.3
4 lines
No id keywords

이 실례에서 개정판본 2.3 은 방안 2 에서 높은 개정판본번호이다.

날자에 의하여 개정판을 검색하려면 선택항목 -c 를 리용한다.

$ **get -c991129 s.payroll.txt**
1.1
4 lines
No id keywords

날자는 다음과 같은 형식으로 규정되며 시간은 선택적이다.

YY [MM [DD [HH [MM [SS]]]]]

sccs

지령 sccs는 지령 get의 기능을 단순화한다.
가장 최근의 개정판본에 대한 읽기전용복사파일을 검색하려면 다음과 같이 한다.

$ **sccs get payroll.txt**

개정판본 1.4에 대한 읽기전용복사파일을 검색하려면 다음과 같이 한다.

$ **sccs get -r1.4 payroll.txt**

payroll.txt의 가장 최근의 개정판본을 편집하려면 다음과 같이 한다.

$ **sccs edit payroll.txt**

지령 sccs는 지령 get의 기발들을 접수한다. 또한 파일이름도 접수하므로 사용자는 "s."으로 시작되는 SCCS파일을 규정하지 말아야 한다. 지령 sccs는 변환을 수행한다.

unget

이 지령은 get -e 혹은 편집지령을 취소함으로써 그 파일에 대한 새로운 증분이 창조되는것을 방지하고 그 파일과 관련된 자물쇠를 제거한다.

delta

지령 delta는 편집된 파일들을 제출하기 위하여 리용되며 새로운 개정판으로 SCCS에로 귀환하기 위하여 필요하다. 이것을 **증분의 창조**(Creating a Delta) 혹은 **파일의 제출**(Submitting a File)이라고 부른다. 파일을 제출하기 위해서는 다음과 같은 지령을 리용한다.

$ **delta s.payroll.txt**
comments? Add fifth line
No id keywords (cm7)
1.5 1 inserted
0 deleted
3 unchanged

지령 delta는 사용자에게 설명문을 요구하는 대화식통보문을 내보낸다. 그것이 입구되면 지령 delta는 그뒤에 새로운 개정판본번호, 삽입된 행의 개수, 제거된 행의 개수 그리고 변화가 없는 행의 개수를 현시한다. 지령 delta는 표준적으로 파일의 쓰기가능한 판본을 제거한다. 만일 기발 -n이 리용되면 쓰기가능한 판본은 그대로 남아 있는다. 한개이상의 개정판을 편집하려면 기발 -r를 리용하여 그 파일이 어느 개정판본경로에 결합되여야 하는가를 규정할수 있다. 지령 sccs는 delta와 함께 리용된다.

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 좀 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 주는 지령들은 가장 우수한 기준으로 된다.

SCCS

sccs – 원천코드조종체계.

---

sccs(1)

이름

sccs – SCCS 지령들을 위한 말단봉사프로그람이다.

형식

sccs [-r] [-d rootpath] [-p dirpath] command [options] [file …]

해설

지령 sccs 는 원천코드조종체계를 구성하는 여러가지 프로그람들에 대한 직접적인 말단지령이다. 이 지령은 SCCS 파일에 대한 공유접근을 허용하면서 사용자설정 ID 를 다른 사용자에게 수행시키는 능력도 가지고 있다. sccs 는 SCCS 파일이름에 대하여 반드시 명백하게 참조되여야 한다는 요구성도 감소시키고 있다. SCCS 파일이름들은 미리 예견된 문자렬 SCCS /s 에 의하여 규정된 작업파일에 생성된다. SCCS 보조등록부의 기정이름은 선택항목 -p 에 의하여 무시될수 있다.

sccs 에 제공되는 지령들은 SCCS 프로그람이거나 의사명령일수 있다. sccs 프로그람들은 admin, cdc, comb, delta, get, help, prs, rmdel, sact, unget, val, what, sccsdiff 를 포함한다. 의사명령들은 다음과 같다:

check 편집되는 모든 파일들의 목록을 현시한다. 파일이 편집되고 있으면 0 이 아닌 값을 귀환한다. 목적은 방안이 설치되기전에 SCCS 파일안에 포함되여 있는 모든것을 검사하기 위하여 프로젝트파일에서 설치항목을 허용하는데 있다(의사명령 info 의 선택항목 -b, -u, -U 에 대한 해설을 참고).

clean 현재등록부 혹은 SCCS 파일로부터 다시 창조될수 있는 등록부로부터 모든 파일들을 제거한다. 편집과정에 있는 파일들은 제거하지 않는다. 선택항목 -b 가 주어 질 때 어느 파일을 편집하겠는가를 결정하는데서 가지(즉 3 개 이상의 성분들을 가진 SID) 들은 무시된다. 결국 가지에서의 편집은 잃게 될수 있다.

create	파일의 내용을 취하면서 초기의 SCCS 파일을 창조한다. 임의의 선택항목들을 접수한다. 만일 파일이 성과적으로 창조되면 초기의 파일들은 앞에 반점을 가지도록 이름이 바뀐다. 읽기전용복사파일들은 get 에 의하여 검색된다. 이름이 바뀐 파일들은 SCCS 파일들이 성과적으로 창조되였는가를 다 검사한후에 제거되여야 한다.

delget	주어 진 파일에 대한 증분을 수행시키고 새로운 판본을 얻는다. 파일에 대한 새로운 판본은 확장된 식별열쇠를 가지며 편집될수 없다. 선택항목 [-mprsy]들은 증분에 통과되며 선택항목 [-bceiklsx]들은 get 에 통과된다.

deledit	get 구문에서 선택항목 -e 를 포함한다는것을 제외하고는 delget 와 동등하다.

diffs	편집되고 있는 파일의 현재판본과 SCCS 형식안에 있는 판본들사이의 차이나는 현시를 준다. 선택항목 [-rcixt]가 get 에 통과된다. 선택항목 [-lsefhb]가 diff 에 통과된다. 선택항목 -C 는 c 로서 diff 에 통과된다.

edit	get -e 와 동등하다.

enter	get 가 생략된다는것을 제외하고는 create 와 동등하다. 이 의사명령은 SCCS 파일을 창조한후에 edit 지령을 즉시에 수행시키려고 할 때 효과적이다.

fix	현재등록부에 있는 증분의 복사판만을 남기고 모든 증분들을 제거한다. 선택항목 -r SID 가 필요하며 원천나무의 잎사귀를 지적하여야 한다. 변경내용에 대한 기록을 하지 않기때문에 fix 를 주의하여 사용하여야 한다.

info	편집되고 있는 모든 파일들을 현시한다. 선택항목 -b 는 어느 파일이 편집되고 있는가를 결정하는데서 가지를 무시한다. 선택항목 -u user 는 user 에 의하여 편집되고 있는 파일들만을 현시한다. 선택항목 -U 는 선택항목 -u current_user 와 동등하다.

print	주어 진 파일들에 대한 정보를 인쇄한다. get -p -m -s 가 뒤에 오는 prs -a 와 동등하다.

tell	매행뒤에 새행문자를 주면서 편집되고 있는 모든 파일을 현시한다(info 의 -b, -u, -U 에 대한 설명을 참고).

unedit	unget 와 동등하다. 마지막 get 로부터 만들어 진 변화는 다 잃게 된다.

일부 지령들인 admin, cdc, check, clean, diffs, info, rmdel, sccs, diff, tell 은 사용자설정 ID 기능을 리용할수 없다. 왜냐하면 이 지령들이 임의의 사람들이 저작권을

리용할수 있도록 변경하기때문이다. 이 지령들은 항상 실지 사용자로서 수행된다.

선택항목들

SCCS 지령에 제공되는 선택항목들은 대응하는 SCCS 안내에 들어 있다. 의사명령에 제공되는 선택항목들은 우의 항목에서 취급하였다. 지령앞에 놓일수 있는 모든 선택항목들은 아래와 같다 :

-r
: 사용자설정 ID 로 되여 있는 사용자가 아니라 실지 사용자로서 sccs 를 수행시킨다.

-d rootpath
: SCCS 파일에 대한 뿌리등록부로 리용될 경로이름을 준다. rootpath 는 표준적으로 현재등록부이다. 이 기발은 환경변수 PROJECTDIR 보다 우선권을 가진다.

-p dirpath
: SCCS 파일에 대한 경로이름을 제공한다. 기정값은 SCCS 등록부이다. dirpath 는 rootpath 에 추가되며 경로이름의 마지막 성분앞에 삽입된다.

지령 sccs -d /usr -p cmd get src/b 는 get /usr/src/cmd/s.b. 로 변환된다. 이것은 별명을 창조할 때 리용될수 있다. 실례로 지령 alias syssccs = "sccs -p/usr/src/cmd" 는 syssccs 를 syssccs get b 와 같은 지령에서 사용되는 별명으로 만든다.

외부적영향

환경변수들

환경변수 PROJECTDIR 가 설정되면 rootpath 에 대한 선택항목 -d rootpath 를 결정하는데 리용된다. 만일 rootpath 가 " / "으로 시작되면 그 값이 직접 리용되고 그렇지 않으면 원천코드 혹은 src 라는 보조등록부에 의하여 검증된 가입이름에 대응되는 가입이름과 홈등록부로 규정된다. 그것이 있다면 이 등록부경로가 사용되고 그렇지 않으면 그 값은 련관된 경로이름으로 된다.

LC_CTYPE 는 본문의 단일/다중바이트문자들에 대한 해석을 결정한다.

LC_MESSAGES 는 현시되는 통보의 언어를 결정한다.

만일 환경변수 LC_CTYPE 혹은 LC_MESSAGES 가 결정되지 않았거나 빈 기호렬로 설정되여 있으면 매 변수에 대하여 LANG 의 값이 기정값으로 리용된다. LANG 이 결정되지 않았거나 빈 기호렬로 설정되여 있으면 LANG 대신에 "C"가 기정값으로 리용된다(lang(5)를 참고).

어떤 국제화변수가 틀린 설정을 포함하면 sccs 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

국제적인 코드모임보장

단일바이트 및 다중바이트기호코드모임이 보장된다.

실례

새로운 SCCS 파일을 창조하려면 다음과 같이 한다.

```
sccs create file
```

한개의 편집파일을 얻어 그것을 편집하며 새로운 증분을 창조하고 편집파일을 얻으려면 다음과 같이 한다.

```
sccs edit file.c
ex file.c
sccs deledit file.c
```

다른 등록부 /usr/src/cmd/SCCS/s.cc.c 로부터 파일을 얻기 위해서는 다음과 같이 하여야 한다.

```
sccs -d /usr/src get cmd/cc.c
```

현재등록부에 있는 모든 파일들의 증분을 만들기 위하여서는 다음과 같이 입력한다.

```
sccs delta *.c
```

가지를 치지 않고 편집된 파일들을 검색하려면 다음과 같이 입력하면 된다.

```
sccs info -b
```

자기가 편집한 파일들의 목록을 얻으려면 다음과 같이 한다.

```
sccs tell -u
```

이미 존재하지 않는 SCCS 파일로부터 원천파일을 얻으려면 다음과 같이 하여야 한다.

```
SRCS = <list of source files>
$ (SRCS) :
        sccs get $ (REL) $@
```

귀환값

성과적으로 완성되면 0 을 귀환한다. 오유가 발생하면 sccs 는 <sysexits.h>로부터의 값을 가지든가 혹은 호출된 지령의 탈퇴값을 가진다. 례외로 되는것은 파일이 편집되고 있을 때 0 이 아닌 상태를 귀환하는 의사명령 check 뿐이다.

관련항목

admin(1), cdc(1), comb(1), delta(1), get(1), prs(1), rmdel(1), sact(1), sccsdiff(1), sccshelp(1), unget(1), val(1), vc(1), what(1), sccsfile(4)

표준일치

sccs: XPG4

제２１장. C와 C++에 대한 개괄

이 장에서는 서로 밀접히 련관된 프로그람작성언어들인 C와 C++에 대하여 소개한다. 이 언어들로 작성한 프로그람들은 실행하기전에 콤파일되여야 한다. 이 장에서는 C와 C++의 발전력사에 대하여 고찰하고 원천코드로부터 실행파일이 어떻게 창조되는가에 대하여 보기로 한다.

C와 C++에 대한 력사

C와 C++의 력사는 C언어로부터 시작된다. C언어는 1970년대 초에 AT&T Bell 연구소에서 Dennis Ritchie에 의하여 처음으로 개발되였다. 이 언어를 만든 기본동기는 연구소의 프로그람작성자들이 새로 나온 DEC(Digital Equipment Corporation)콤퓨터를 위한 연산체계 UNIX를 쓸수 있도록 하기 위한것이였다. 1970년대 후반기에 C콤파일러들이 상업적인 목적에 리용되면서 C언어는 인기 있는 언어로 장성하기 시작하였다.

C언어의 보편성이 커지면서 더욱더 많은 판매자들이 자기의 고유한 콤파일러들을 판매하기 시작하였다. 그런데 C언어정의의 몇가지 모호성으로 하여 판매자들은 C언어를 자기식의 관점에서 마구 개작하게 되였으며 이것은 언어의 분렬을 초래하게 하였다. 이로부터 C프로그람을 짜는 프로그람작성자들은 자기들의 원천파일이 어느 C콤파일러로 콤파일될수 있는가 하는것을 담보하지 못하였다.

1980년대 초에 ANSI(American National Standards Institute)는 C언어에 대한 정의를 표준화하였다. 결과 ANSI C콤파일러로 프로그람을 콤파일한 프로그람작성자들은 누구나 다 자기의 프로그람이 ANSI C콤파일러가 갖추어 져 있는 임의의 체계에서 아무런 수정이 없이 콤파일될수 있다는 확신을 가지게 되였다.

C++언어는 AT&T Bell 연구소의 Bjorne Stroestrup에 의하여 개발되였다. C++라는 이름은 그것이 C언어에 토대한 개선된 언어라는 관점에서 1983년에 Rich Mascitti가 달아준것이다. C프로그람들도 C++콤파일러에 의하여 정확히 콤파일된다. 이것은 C++에서 아직 순차적으로 프로그람을 작성하여야 한다는것을 의미한다.

언어전체를 처음부터 다시 설계하지 않고 토대언어로서 C를 선택한 리유는 새 언어설계가 C언어에서 그러했던것처럼 많은 문제들이 산생될수 있기때문이며 C언어에서 제기되였던 문제들이 어느정도 알려 져 있다는것과 관련된다.

C++의 초기방안들은 "클라스를 가진 C"라고 알려 져 있었으나 세월이 흐르면서 오늘의 언어로 발전하여 왔다. 클라스의 실현이 C++가 C와 차이나게 하는 기능의 전부인것은 아니다. C++에는 이와 함께 연산자의 재정의, 참조, 가상함수와 같은 다른 기능들도 있다. 1989년에 Hewlett-Packard의 창안으로 ANSI는 C++의 표준화를 하기 시작하였다. 오늘날 ANSI C++규격은 널리 리용되고 있다.

C 와 C++의 콤파일러

콤파일러는 언어의 원천코드본문을 콤퓨터가 읽을수 있는 기계어로 변환하기 위하여 설계된 복잡한 프로그람이라는것밖에는 아무것도 아니다. 콤파일러는 특정한 콤퓨터환경에서 특정한 콤퓨터언어를 위하여 씌여 진다. 즉 IBM 체계에서 동작하도록 씌여 진 콤파일러는 Sun 체계를 위하여 씌여 진 콤파일러와 다른 프로그람으로 된다.

콤파일된 프로그람은 먼저 콤퓨터체계의 파일에 들어 간다. 대부분의 프로그람작성자들이 편집기 vi, ed, emacs 를 사용한다. 원천코드를 포함하는 파일은 원천파일로 고찰된다. 콤파일러에 따라 C 의 원천파일은 ".c"로 끝나며 C++의 원천파일은 ".cc" 혹은 ".cpp"로 끝난다. C 원천파일이름의 정확한 실례는 paroll.c 이고 C++의 원천파일이름의 정확한 실례는 paroll.cc 혹은 payroll.cpp 이다.

원천파일안에 있는 프로그람을 보통 원천코드 혹은 **원천프로그람**(Source Program)이라고 부른다. 원천코드는 원천파일에 넣어 진후에 콤파일될수 있다. 단순하게 고찰하기 위하여 파일확장자 ".c"를 리용하기로 한다. 그러나 이것은 ".cc"나 ".cpp"로 바뀔수 있다.

프로그람의 콤파일

콤파일작업은 원천파일을 지적하는 지령을 사용함으로써 시작할수 있다.

ANSI C 콤파일러들은 콤파일지령의 앞에 "a"를 붙여 콤파일을 시작할수 있다. 한편 C++콤파일러에 대해서는 "a"대신에 "CC"를 리용한다. 이 책에서는 단순성을 위하여 지령 cc 를 리용한다. 정확한 지령은 콤파일러쏘프트웨어를 참고하면 된다. 프로그람작성자가 콤파일지령을 입장시킬 때 체계는 **전처리기**(Preprocessor), **콤파일러**(Compiler), **아쎔블러**(Assembler), **련결편집기**(Link Editor)와 같은 4 가지 성분들을 거치게 된다.

콤파일러의 전반과정은 다음과 같은 두개의 기본적인 단계들로 설명할수 있다.

① 전처리기, 콤파일러, 아쎔블러로 이루어 진 콤파일과정

② 보다 복잡한 프로그람들을 위한 련결과정. 여기서 객체파일들은 하나의 실행프로그람의 형태로 서로 련결된다.

이 과정을 그림 21-1 에서 보여 주고 있다.

원천프로그람은 콤파일되기전에 전처리기를 거쳐야 한다. 이 과정은 자동적으로 진행되며 전처리기는 콤파일러를 위한 원천코드를 준비한다. 실례로 콤파일러를 위하여 포함되여야 할 정보를 발생하는 **전처리기지령**(Preprocessor Directive)들이 있다. 이에 대해서는 앞으로 C 와 C++ 프로그람작성법을 론의할 때 더 설명하기로 한다.

콤파일러는 원천파일에 있는 프로그람작성명령문들에 대응하는 아쎔블러언어를 창조한다.

아쎔블러는 기계가 읽을수 있는 객체코드를 생성한다. 매 원천파일에 대하여 한개의 **객체파일**(Object File)이 창조된다. 매 객체파일은 앞붙이를 제외하고는 원천파일과 같은 이름을 가지며 확장자는 ".o"로 교체된다.

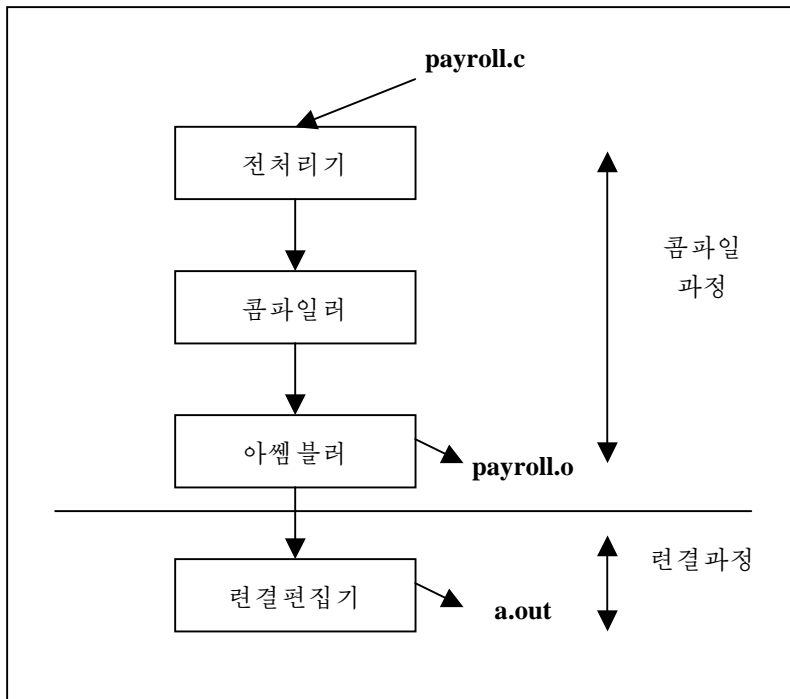



그림 21-1. 간단한 실행과정

프로그람작성자는 한개의 프로그람안에서 모든 함수들을 반드시 정의해야 하는것은 아니다. 어떤 함수들은 서고안에 미리 씌여 져 있다. 콤파일러의 판매자는 파일에 대한 읽기, 쓰기 그리고 수학적함수 등과 같은 기본함수들을 위한 몇개의 서고들을 제공하여 준다. 쏘프트웨어회사들에서 보다 많은 서고들을 제공할수도 있으며 프로그람작성자들이 자기식의 서고들을 창조할수도 있다.

최종단계에서 련결편집자는 프로그람에서 리용된 함수들이 있는 서고들을 찾고 실행가능한 파일을 창조하기 위하여 프로그람작성자의 객체파일과 서고의 객체파일들을 결합한다.

프로그람을 위한 이 모든 단계의 콤파일과정들이 성과적으로 완성되면 **실행가능한 파일**(Executable File:보통 실행파일이라고 부른다.)이 창조되며 객체파일은 자동적으로 제거된다. 기정값으로 제공되는 실행가능한 파일의 이름은 a.out이다.

프로그람을 실행시키려면 실행파일의 이름을 입력하여야 한다. 이 경우에 실행파일의 이름은 a.out이다.

프로그람을 콤파일하기 위해서는 다음과 같이 지령을 입력하여야 한다.

```
$ cc payroll.c
```

콤파일러의 선택항목

우에서 이미 고찰한것처럼 콤파일러들은 한개의 원천파일을 콤파일할수 있다. 그러나 어떤 경우에는 프로그람이 여러개의 원천파일안에 나뉘여 있을수 있다. 이러한 배치는 큰 프로그람을 작성할 때 혹은 프로그람작성을 설계할 때 아주 편리하다.

여러개의 원천파일들을 콤파일하기 위해서는 다음과 같이 하여야 한다.

$ **cc payroll_1.c payroll_2.c employee.c**

기정값으로 제공되는 실행파일의 이름 a.out 는 의미가 전혀 반영되지 않은 이름이다. 선택항목 -o 는 프로그람작성자들이 생성되는 실행파일의 이름을 규정할수 있게 해준다. 실례로 3 개의 원천코드파일들이 생활비계산프로그람의 코드들을 모두 포함하고 myPayroll 이라는 실행파일을 필요로 한다면 다음과 같은 지령을 리용하여야 한다.

$ **cc -o myPayroll payroll_1.c payroll_2.c employee.c**

그러면 myPayroll 라는 이름을 가진 실행파일이 콤파일되며 myPayroll 프로그람을 실행시키려면 다음과 같은 지령을 주어야 한다**.**

$ **myPayroll**

만일 프로그람작성자들이 콤파일을 련결과정의 앞에서 멈출것을 요구한다면 선택항목 -c 를 리용한다. 선택항목 -c 를 리용하면 실행파일은 생성되지 않지만 사용자들이 객체파일을 볼수 있게 된다. 다음 실례의 경우에는 원천파일 payroll_1 과 payroll_2 에 대응하는 객체파일들은 있지만 실행파일은 생성되지 않는다.

$ **cc -c payroll_1.c payroll_2.c**

원천코드에 대한 모든 객체파일들이 만들어 진 다음에는 프로그람작성자들이 실행파일을 생성할수 있게 되며 콤파일지령에서 원천파일대신에 객체파일을 사용할수 있다.

$ **cc -o myPayroll payroll_1.o payroll_2.o employee.o**

콤파일러는 ".o"라는 파일확장자를 식별하고 그 파일들이 련결단계에만 필요하다는 것을 식별한다. 확장자 ".c"와 ".o"를 둘 다 같은 지령행에 포함시킬수도 있다. 그때 ".o" 파일들은 련결되기만 하고 ".c"파일들은 콤파일되고 련결된다. 콤파일러는 또한 ".s"로 끝나는 아쎔블러파일들도 접수하고 그것을 조립하고 련결하면서 그것을 적당히 조종한다.

서로 다른 원천파일들을 많이 가지고 있는 큰 프로그람에서 작업할 때 그것들을 개별적으로 콤파일하고 후에 함께 련결하면 시간을 절약하게 된다. 만일 어떤 한개의 원천파일만 외부적영향을 많이 받았다면 콤파일러는 그것만 콤파일하여 다른것들과 련결시킨다.

어떤 파일들을 콤파일해야 하는가에 대한것은 make 봉사프로그람을 리용하면 자동화할수 있다.

C 및 C++의 make 봉사프로그람

앞절에서는 콤파일이 두개의 기본적인 단계 즉 콤파일과 련결을 통하여 진행되여야 한다는것을 보았다. make 봉사프로그람은 개발생명주기를 통하여 이러한 과정을 관리한다. 이 절에서는 make 봉사프로그람을 고찰하는데 봉사프로그람은 UNIX 의 변종에 따라 얼마간 변한다는데 주의하여야 한다.

make 봉사프로그람으로 여러개의 원천파일들로 구성된 프로그람을 관리하면 대단히 효과적이다. 이것은 틀에 박히지 않고 본보기파일을 리용하는 위력한 방법으로 파일들사이의 호상종속성모임을 유지하고 있다.

실례로 make 는 C, C++ 혹은 HTML 원천코드에 대한 변경을 관리하는데 리용될수 있다. 객체파일과 실행파일사이의 관계는 make 봉사프로그람을 통하여 서술되며 그것은 후에 작성자가 변경할 때 갱신된다. 물론 make 는 쉘로부터 직접 호출시키거나 지령행으로부터 수행시킬수 있지만 복잡한 구축과정은 보통 제작파일이라고 부르는 파일에서 진행된다.

Makefiles

제작파일(Makefile)은 다음의 4 가지 형태의 행들중에서 임의의것을 포함하는 본문파일이다. 즉 **목적행**(Target Line), **쉘지령행**(Shell Command Line), **마크로행**(Macro Line), **Make 지령행**(Make Directive Line)들중의 하나를 포함한다. 설명문들도 제작파일에 포함될수 있으며 기호 "#"로 시작된다.

제작파일을 기동시키려면 프로그람작성자는 make 를 입력한다. 지령 make 는 현재작업등록부에서 makefile 이라는 이름을 가진 파일을 찾는다. 만일 makefile 이 없다면 make 는 Makefile 이라는 이름을 가진 파일을 탐색한다. make 봉사프로그람은 이 장의 뒤부분에 있는 《지령행으로부터 make 의 실행》에서 고찰되는 선택항목들을 통하여 기정값과는 다른 이름들도 탐색할수 있다. 만일 makefile 혹은 Makefile 이 지령 make 를 포함하는 파일의 파일이름으로 리용되지 않았으면 습관적으로 확장자 ".mk"가 붙는다.

목적파일과 종속성파일

목적행들은 무엇을 구축할수 있는가를 지적한다. 목적행들은 순차적으로 놓인 목적파일목록, 두점(:), 종속성 혹은 필수파일목록들로 구성된다.

목적파일은 한개이상의 **필수파일**(Prerequisite File)에 의존하는 파일이다. 필수파일은 **종속성**(Dependency)이라고 부르기도 한다. 만일 종속성이 목적파일보다 더 최근에 갱신되였다면 make 는 목적행의 뒤에 있는 **구축지령**(Construction Command)에 기초하여 목적파일을 갱신한다. make 봉사프로그람은 구축과정에 오유가 발생하면 실행을 멈춘다. 간단한 제작파일은 다음과 같은 형식을 가진다.

```
target:dependencies_list
        construction_commands
```

구축지령은 보통 콤파일하고 련결하거나 혹은 련결만 함으로써 목적파일을 구축하는 쉘의 표준지령이다. 다음의 코드는 payroll_form 이라는 파일의 구축지령들을 포함하는 목적행을 보여 준다. 그 파일의 종속성파일은 form.o 와 pay_info.o 이다. 지령 cc 는 목적파일 payroll_form:form.o pay_info.o 를 구축한다.

```
payroll_form:form.o  pay_info.o
cc  -o  payroll_form:form.o  pay_info.o
```

목적파일목록은 여러개의 목적들을 포함할수 있으나 대체로 한 목적행에 한개의 목적파일만 놓인다. 목적파일목록은 빌수 없지만 종속성목록은 빌수 있다. 아래에서는 목적행의 실례들을 보여 준다.

```
target_1:dependency_1  dependency_2
                        # 종속성이 두개인 목적행
target_2:               # 종속성이 없는 목적행
```

종속성들은 총체적인 실행파일이 조성되기전에 일정한 성분들이 구축되도록 담보하기 위하여 리용된다. 목적파일은 그것의 종속성보다 새 방안으로 되여야 한다. 만일 어떤 종속성이 현재의 목적파일보다 새것이거나 종속성이 존재하지 않으면 그 종속성이 만들어 지고 다음에 현재의 목적파일이 구성되여야 한다. 만일 종속성목록이 비여 있으면 그것의 목적파일은 항상 조성된다.

종속성그라프를 구축할수 있으며 거기서 머리부파일들에 대한 종속성들도 볼수 있다. 여러개의 기호적상수들과 마크정의들이 프로그람의 다른 원천들에서 리용된다면 그것들을 **머리부파일**(Header File) 혹은 **포함파일**(Include File)이라고 하는 하나의 파일에 모아 놓을수 있다. 이 파일들은 .h 로 끝난다.

기호적상수는 원천프로그람에서 상수값의 위치에서 리용할수 있는 이름이다. 실례로 만일 생활비계산프로그람에서 자동차의 마일당 운임을 위한 종업원의 통근비을 계산하는 것이 필요하다면 마일당가격을 기호상수 AUTO_MILEAGE 에 련관시킬수 있다.

```
# define  AUTO_MILEAGE  .33
```

이 가격이 필요한 임의의 곳에서 프로그람작성자는 실제값 $.33 대신에 AUTO_MILEAGE 를 리용할수 있다. 마일당 통근비가 변경되면 기호상수값은 머리부파일에서만 변경시키면 된다. 그러면 원천파일에 머리부파일을 포함시키고 제작파일에는 종속성으로서 포함시키면 프로그람안의 어디서나 AUTO_MILEAGE 의 리용은 변경될것이다.

마크로들은 짧은 함수와 류사한것으로서 역시 머리부파일에 포함될수 있다. 실례로 만일 한 종업원의 생활비 총액이 감소된후에 프로그람전반에서 그것의 값이 0 보다 작지 않도록 담보하는 간단한 함수가 필요하다면 마크로로서 그 함수를 정의하고 그 마크로를 리용하는 여러 원천파일들에 대한 머리부파일에 포함시킬수 있다.

```
# define   iswagevalid(n)  (n>=0)
```

그림 21-2 에서 수행파일 payroll_from 은 두개의 객체파일들에 종속되고 매 객체파일들은 각각 한개의 원천파일들에 종속되며 머리부파일 from.h 에 종속된다. 또한 from.h 는 다른 두개의 머리부파일에 의존한다.

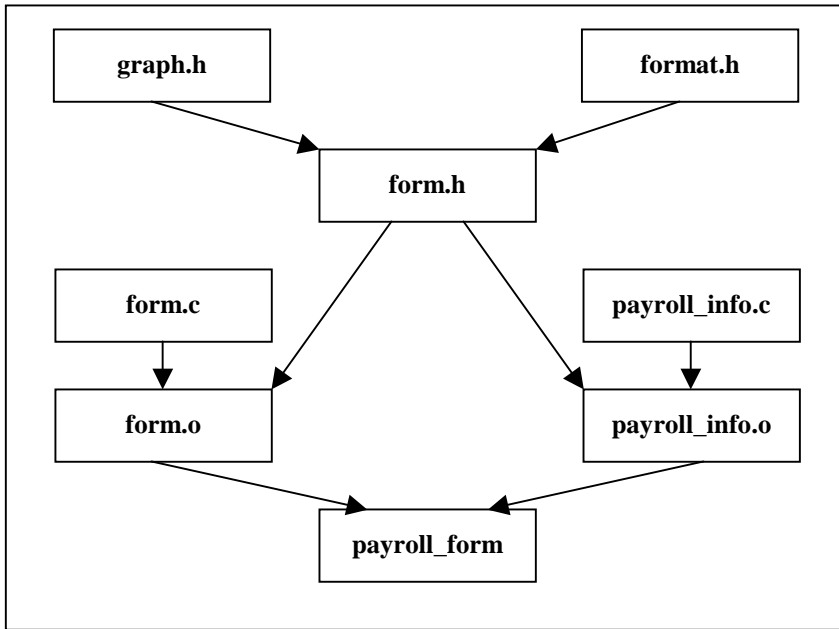



그림 21-2. 종속성

매 종속성들은 다른 종속성에 대한 목적파일로 될수 있다. 실례로 두개의 from.o 와 payroll_info.o 는 한개의 행에서는 목적파일로 되지만 다른 행에서는 다른 목적파일에 대한 종속성으로 된다. 겹친 종속성들은 많은 파일들사이의 호상관계를 규정하는 복잡한 계층구조를 조성하게 된다. 아래에 제시된 제작파일은 우에서 본 완전한 종속성도표에 대응한다.

```
payroll-form:  form.o  payroll-info.o
cc  -o  payroll_form  form.o  payroll_info.o

form.o:  size.c  form.h
cc  -c  size.c

payroll_info.o:  payroll_info.c  form.h
cc  -c  payroll_info.c

form.h:  graph.h  format.h
cat  graph.h  format.h  >  form.h
```

마지막행은 어떻게 쉘지령들이 구축행에 놓일수 있는가를 보여 주고 있다.

서고목적파일

서고들은 여러개의 함수 혹은 클라스들로 구성된다. 함수 혹은 클라스들은 보통 론리적으로 그룹화되며 여러 프로그람들에서의 재리용을 위하여 서술한 이름으로 서고에 배치된다. 만일 목적파일 혹은 종속성파일이 괄호를 가진다면 그것은 서고로 고찰된다.

서고목적파일을 리용하는 목적행의 실례는 다음과 같다.

```
payroll_lib.a(payroll_tools.o): payroll_toos.c  payroll.h
```

괄호들사이에서의 공백은 오유로 되기때문에 사용하지 말아야 한다.

규칙목적

make 의 위력한 특징은 뒤붙이규칙, 추론규칙과 같은 일반적인 규칙들을 규정하는 능력에 있다. 규칙들은 일정한 형태의 목적파일들을 어떻게 구축하는가를 make 에게 알려 주기 위한 속기방법이다. 아래에서는 제작파일로부터 추려 낸 내용을 보여 주고 있다. (콤파일선택항목 -I 는 머리부파일들을 찾을 경로를 콤파일러에 알려 준다.)

```
payroll.o:        payroll.c
cc  -c  payroll.c  -I.  -I/user/local/include
screens.o:        screens.c
cc  -c  screens.c  -I.  -I/user/local/include
main.o: main.c
cc  -c  main.c  -I.  -I/user/local/include

main:    payroll.o  sdreens.o  main.o
cc  payroll.o  screens.o  main.o
```

우의 내용을 보면 중복되는 내용이 많다. 모든 쉘지령행들은 그 파일이 실행파일로 콤파일된다는것을 제외하고는 동일한것들이다. 다음의 규칙은 .c 로 끝나는 파일을 .o 로 끝나는 파일로 어떻게 변환해야 하는가를 make 에게 알려 준다.

```
.c.o:
          cc  -c  $<  -I.  -I/user/local/include
```

이 규칙은 특수한 마크로 $<를 가지는데 그것은 규칙의 몸체부분에 현재의 원천파일을 대입한다는것을 의미한다. 이 규칙을 리용하면 .c.o 는 제작파일로부터의 발취를 다음과 같이 함축한다.

```
.c.o:

cc  -c  $<  -I.  -I/user/local/include

main:    payroll.o  screens.o  main.o
    cc  payroll.o  screens.o  main.o
```

마크로

마크로는 공통적으로 리용되는 본문에 대한 참조를 단순하게 하며 정보가 한개의 장소에서만 정의되도록 함으로써 프로그람을 효과적으로 유지하고 읽기 쉽게 작성하도록 한다.

제작파일에서 마크로정의의 기본문법형식은 다음과 같다 :

이름 = 값목록

이름은 대소문자, 수자(0 – 9), 그리고 밑선(_)의 임의의 결합으로 만들수 있다. 마크로이름은 습관적으로 모두 대문자로 쓴다. make 봉사프로그람은 이름을 식별이름으로 교체되게 한다.

값목록은 없거나 하나 혹은 여러개의 구체례들을 포함할수 있다. 이름은 파일이름들의 목록으로 교체될수 있다. 값목록은 대단히 길수 있는데 이런 경우에 \을 리용하여 다른 행에서의 정의를 계속하도록 할수 있는데 여기서 \은 한개 공백으로 본다.

마크로정의는 제작파일안에서 자주 리용되는 머리부파일의 그룹을 표현하는데 리용될수 있다.

HEADERS = payroll.h companies.h benefits.h

제작파일은 다음과 같이 HEADERS를 리용할수 있다.

Process.o: $(HEADERS)

지령행으로부터 make의 수행

make 봉사프로그람은 제작파일안에서뿐아니라 지령행의 선택항목들에 의해서도 실행될수 있다. 이 선택항목들은 제작파일을 실행할 때 리용된다.

make의 대표적인 지령행인수들은 임의의 순서로 나타날수 있다.

Make [-f 제작파일] [options] [macro definitions] [targets]

선택항목들은 [ ]안에 포함된다.

지령행선택항목들은 -로 시작되여 그뒤에 선택항목이 놓인다.

아래에서는 보다 일반적으로 사용되는 몇가지 선택항목들을 보여 주고 있다.

make 지령행의 선택항목들

-d	오유검사방식을 취하게 한다. 오유검사방식에서는 많은 량의 통보가 생성된다. 이 방식은 제작파일에 대한 오유처리의 마지막수단으로만 되여야 한다.
-f	제작파일로 사용될 파일의 이름을 규정한다. 선택항목과 파일이름사이에는 공백이 있어야 한다.

-p 모든 마크로정의와 규칙들을 인쇄한다.

-i make는 지령에 의하여 귀환되는 0이 아닌 오유코드들을 무시한다. 모든 목적파일들의 구축은 끝나는것이 아니라 계속된다.

-k 지령에 의하여 0이 아닌 오유코드가 귀환되면 현재 목적파일에 대한 작업을 그만둔다. 다른 목적파일에 대한 작업은 계속된다. 이것은 -s방식의 반대이다.

-s make를 **침묵방식**(Silent Mode)에 놓는다. 표준적으로 지령들은 표준출력장치에서 수행된다.

-t 규칙과 관련된 지령들은 수행되지 않고 파일을 창조만 한다(UNIX의 touch지령을 참고).

C와 C++의 오유검사

프로그람들이 문법오유를 성과적으로 처리하고 콤파일러가 실행파일을 생성하였다면 이제는 프로그람의 실행상태와 론리적오유들은 검사하여야 한다. 오유검사프로그람들은 이러한 오유가 어디에서 나타나는가를 찾도록 도와 준다.

C와 C++콤파일러들은 프로그람에서 허용되는 여러가지 종류의 구조들을 잘 처리할수 있다. 콤파일러는 토막을 침해하는것과 같은 심각한 오유가 나타나면 프로그람의 실행을 정지시킨다. 그리고 Segmentation violation - Core dumped 라는 통보가 현시되며 **핵심파일**(Core File)을 창조한다. 핵심파일은 실패가 일어 났을 때 프로그람이나 체계에 대한 상태정보를 포함한다. **탄창추적**(Stack Trace)은 핵심파일에 들어 있는 정보의 일부분을 보여 준다. 이 추적은 **쏟기**(Dump)가 발생한 원천행과 그 행에 도달할 때까지 호출된 함수들을 지적한다.

sdb(기호적인 오유검사프로그람) 혹은 dbx와 같은 오유검사프로그람들은 프로그람작성자가 문제를 찾고 수정하는것을 돕기 위하여 핵심파일을 리용한다. 오유검사프로그람들인 sdb와 dbx는 매개가 자기의 고유한 특징을 가지지만 이 프로그람들은 류사한 가능성을 제공한다. 그것들은 둘 다 프로그람작성자들이 프로그람의 실행을 조종감시할수 있게 해준다. 프로그람은 실행환경에 대한 상태검사를 하면서 한 행씩 수행시킬수 있다. 또한 이것들은 핵심파일도 검사한다. 오유검사프로그람은 중단점을 통하여 어느 행에서 프로그람을 림시로 정지시키겠는가를 조종할수도 있다. 실행이 림시로 정지되면 이때 프로그람작성자는 변수들과 구조체들안에 있는 값들과 체계에 대한 검사를 진행할수 있다. 프로그람작성자는 실행을 한 걸음씩 지켜 볼수 있으며 어떤 점에서부터 계속해 나갈수도 있다.

lint는 또 다른 하나의 유용한 도구이다. 코드는 완전히 명백하게 콤파일될수 있지만 부정확하게 수행될수 있다. 지령 lint는 문제점이 발생할수 있는 곳에서 원천코드를 검사한다. lint는 값이 대입되기도 전에 사용된 변수들, 리용되지 않은 함수의 인수들, 귀환할수 없는 귀환값을 사용하는 함수들을 포함하여 광범한 범위의 문제점들과 잠재적인 문제점들을 검사하고 통보하므로 콤파일러보다 더 엄밀하다.

lint 가 내보내는 경고를 무시하고 콤파일을 완성시키지 않을수도 있지만 그 경고는 원칙적으로 프로그람에 오유가 있거나 이동불가능한 구조를 사용하고 있는데 표준적인 프로그람작성법을 위반하였다는것을 의미한다. 따라서 lint 의 경고에 주의를 돌리는것은 프로그람의 오유검사만이 아니라 프로그람작성기술을 높이는데서도 좋은 방법으로 된다.

지령소개

아래에서는 이 장에서 리용한 지령들을 묶어서 보여 주고 있다. UNIX 변종들에서 지령은 약간 차이나기때문에 지령들의 선택항목이나 다른 부분들에서 일부 차이나는 점들을 볼수 있을것이다. 그러나 아래에서 제시하는 지령들은 가장 우수한 기준으로 된다.

make

make – 파일들의 그룹을 관리, 갱신, 재생성한다.

---

make(1)

이름

make – 프로그람들의 그룹을 관리, 갱신, 재생성한다.

형식

make [-f 제작파일] [-bBdeiknpqrsSt] [macro_name = value] [names]

해설

제작파일의 구조

제작파일은 목적행, 쉘지령행, 마크로정의, 포함행과 같은 4 개의 서로 다른 행들을 포함할수 있다.

목적행 :

목적행은 순차적으로 놓인 공백분리기호, 비지 않은 목적파일들의 목록, : 혹은 ::, 종속성이라고 부르는 필수파일목록(비여 있을수도 있다.)들로 구성된다. 종속성과 같은 파일이름들의 생성에서는 패턴정합표시법(regexp(5)를 참고)이 제공된다.

쉘지령행 :

목적행에서 ;이뒤에 있는 본문과 태브로 시작하는 모든 행들은 목적파일을 갱신하기 위하여 실행하여야 할 쉘지령이다(아래에서 환경변수 SHELL 에 대한 부분을 참고).

태브 혹은 #로 시작하지 않는 첫 행은 새로운 목적정의, 마크로정의, 포함행의 시작으로 된다. 쉘지령들은 < \ >, 새 행문자렬들을 리용하여 여러개 행들로 계속될수 있다.
지령행과 관련된 목적행들을 **규칙**이라고 부른다.
마크로들 :

string1 = string2 형식의 행들은 마크로정의이다. 마크로들은 제작파일의 임의의 곳에서 정의될수 있다. 그러나 보통 앞부분에서 그룹화된다. string1 은 마크로이름이고 string2 는 마크로의 값이다. string2 는 설명문자 혹은 새 행문자까지의 모든 문자들로서 정의된다. "="의 왼쪽과 오른쪽에 있는 공백과 태브들은 무시된다.
제작파일안의 임의의 곳에서 나타나는 $ (string1)은 string2 로 교체된다. 만일 마크로이름이 한개 문자이든가 대입되는 값이 없으면 괄호는 생략될수 있다. 선택적대입렬인 $ (string1 [:subst1 = [subst2]])이 규정되면 그것은 string1 안에서 subst1 이 겹치지 않고 나타날 때마다 subst1 을 subst2 로 교체한다. 마크로값안에 있는 부분문자렬들의 분리는 공백, 태브, 새행문자, 행시작 등으로 진행된다. 만일 OBJS = file1.o file2.o file3.o 에 대하여 $ (OBJS: .o = .c)을 실행하면 file1.c file2.c file3.c 로 된다.

마크로값들은 또 다른 마크로에 대한 참조를 포함할수 있다.

```
ONE = 1
TWELVE = $ (ONE) 2
```

$ (TWELVE)의 값은 $(ONE)2 로 설정되여 있지만 그것이 목적파일, 지령, 포함행에서 리용될 때에는 12 로 바뀐다.

마크로정의는 지령행에서 규정될수도 있으며 제작파일에 있는 임의의 정의를 무시할수도 있다.

(XPG4 의 경우에 지령행에 있는 마크로는 MAKEFLAGS 라는 환경변수에 추가된다. 이때 환경변수 MAKEFLAGS 에서 정의된 마크로는 동일한 이름을 가지고 이미 존재하는 환경변수를 무시하면서 환경에 추가된다.)

일정한 마크로들은 make 를 위하여 자동적으로 정의된다. 마크로정의가 취급되는 순서에 대한 론의는 환경변수부분을 참고하면 된다.

마크로에 대입된 값은 조건적마크로정의에 의하여 무시될수 있다. 조건적마크로정의는 target := string 1 = string2 의 형식을 가진다. 목적행이 처리되고 있을 때 조건적마크로를 조성할 때 규정된 마크로값이 효과를 나타내게 된다. 만일 string1 이 이미 정의되여 있다면 string1 의 새 값은 이전의것을 무시한다. string1 의 새 값은 목적파일 혹은 그것의 종속성이 처리될 때 효과를 가진다.

포함행 :

제작파일에서 행의 첫 7 개 문자가 문자렬 include 로 나타나고 그뒤에 한개이상의 공백 혹은 태브문자들이 뒤따른다면 그 행의 나머지부분은 파일이름으로 가정되며 현재 make 의 입장에 의하여 읽고 처리된다.

일반적해설

make 는 제작파일안에 미리 들어 있는 지령들을 수행하여 한개이상의 목적파일들을 갱신한다. 목적파일의 이름은 원칙적으로 프로그람의 이름과 같다. 만일 선택항목 -f 가 규정되지 않으면 파일이름들은 제작파일, 제작파일, s.제작파일, SCCS/s.제작파일, s.제작파일, SCCS/s 제작파일이 순서대로 취해 진다. -f -가 규정되면 표준입구장치장치가 리용된다. 한개이상의 -f 가 규정될수 있다. -f 와 파일이름사이에는 공백이 있어야 하며 여러개의 파일이름들은 각각 그앞에 선택항목 -f 를 가져야 한다. 제작파일의 내용은 체계의 규칙과 마크로가 나타나도 그것을 무시한다.

만일 지령행에 아무런 목적이름도 규정되여 있지 않으면 make 는 추론규칙이 아닌 첫번째 제작파일에 있는 첫번째 목적파일을 갱신한다. 목적파일은 자기보다 더 새로운 파일에 종속될 때에만 갱신된다. 파일이 없으면 날자가 초과된것으로 간주된다. 목적파일의 모든 종속성은 필요하다면 목적파일이 갱신되기전에 먼저 갱신된다. 이것은 목적파일의 종속성나무에 대한 깊이우선갱신으로 된다.

만일 목적파일이 목적행에서 분리기호뒤에서 규정되는 아무런 종속성도 가지지 않는다면(명백한 종속성) 그 목적파일과 관련된 임의의 쉘지령들은 목적파일이 날자초과일 때에만 수행된다.

목적행은 목적파일이름(혹은 이름)들과 명백한 임의의 종속성이름들사이에 한개 혹은 두개의 ：을 가질수 있다. 목적파일이름은 한개이상의 목적행에서 나타날수 있으나 이 모든 행들은 같은 형식(：혹은 ：：)으로 되여야 한다. 한개의 ：을 리용하는 경우에 이 목적행들의 제일 첫 행은 그것과 관련된 명백한 지령들을 가질수 있다. 만일 목적파일이 어떤 행에 있는 어떤 종속성에서 날자초과로 되여 있을 때 명백한 지령들이 규정되였으면 그 지령들이 수행되고 규정되지 않았으면 기정의 규칙이 수행될수 있다. ：：을 리용하는 경우에 명백한 지령들은 목적파일이름을 포함하는 목적행들의 매개와 관련될수 있다. 만일 목적파일이 특수한 행에 있는 어떤 종속성에서 날자초과로 되여 있으면 그 행을 위한 지령들이 수행된다. 체계의 규칙도 수행될수 있다.

목적행들과 그것들과 관련되는 쉘지령행들도 규칙으로 참조될수 있다. ＃와 새행문자는 규칙들을 제외하고는 제작파일의 어디서나 설명문을 포함할수 있다. 규칙에서 설명문은 SHELL 마크로의 설정에 종속된다.

다음의 제작파일은 pgm 이 두개의 파일 a.o 와 b.o 에 종속되고 그것들은 다시 대응하는 원천파일들(a.c 와 b.c)과 공통파일 incl.h 에 종속된다는것을 보여 준다.

```
OBJS = a.o  b.o
pgm: $(OBJS)
        cc $(OBJS) -o pgm
a.o: incl.h  a.c
```

```
        cc -c a.c
b.o: incl.h b.c
        cc -c b.c
```

지령행들은 고유한 쉘에서 한번에 한개씩 수행된다. 매 지령행은 -, @, +와 같은 앞붙이들가운데서 한개 혹은 여러개를 가질수 있다. 이 앞붙이들은 아래에서 설명된다.

0 이 아닌 상태를 귀환하는 지령들은 make 를 종결한다. 선택항목 -i 혹은 제작파일안에서 특정의 목적 .IGNORE 의 출현은 아직 오유통보는 표준출력장치장치로 인쇄되고 있다고 해도 몇개의 지령행들이 오유를 발생시켰는가에 무관계하게 제작파일을 계속 수행하게 한다. 만일 -가 지령행의 시작에 나타나면 그 행에 의하여 귀환된 오유는 언제나 표준출력장치장치로 출력되고 make 는 종결되지 않는다. 앞붙이 -는 제작파일에서 선택적으로 오유분석에 리용될수 있다. 만일 선택항목 -k 가 규정되고 지령행이 오유상태를 귀환한다면 작업은 현재 목적파일에서 취소되고 그 목적파일에 의존하지 않는 다른 가지들은 계속된다. 만일 선택항목 -k 가 환경변수 MAKEFLAGS 에 제출되면 처리는 선택항목 -S 를 규정함으로써 기정값으로 귀환될수 있다.

선택항목 -n 은 실행을 하지 않고 지령행의 출력을 규정한다. 그러나 만일 지령행이 문자렬 $(MAKE) 혹은 ${MAKE} 를 가지거나 앞붙이로서 +를 가진다면 그 행은 항상 실행된다. 선택항목 -t 는 지령을 수행하지 않고 파일의 수정날자를 갱신한다.

지령행은 표준적으로 실행되기전에 인쇄되지만 만일 행의 시작에 @가 있다면 인쇄는 억제된다. 선택항목 -s 혹은 제작파일에서 특정의 목적파일 .SILENT：가 나타나면 모든 지령행들의 인쇄는 금지된다. @는 선택적으로 인쇄를 금지시킨다(초기의 태브를 제외하고). make 로 인쇄한 모든것은 변경없이 직접 쉘에 전달된다. 다음의 지령은 쉘이 하는것처럼 ab 를 생성한다.

```
echo a\
b
```

선택항목 -b 는 낡은 제작파일이 오유없이 수행되도록 한다. make 의 낡은 방안은 만일 목적파일이 그 목적파일과 관련된 임의의 명백한 지령들을 가지지 않으면 사용자는 지령을 0 으로 보고 정의되여 있는 임의의 .DEFAULT 규칙을 수행하지 않는다고 가정하고 있다. make 의 현재 방안은 표준적으로 이 방식에서 연산한다. 그러나 make 의 현재 방안은 만일 목적파일이 그 목적파일과 관련된 명백한 지령들을 가지지 않고 .DEFAULT 규칙이 정의되여 있다면 .DEFAULT 규칙을 수행하도록 하는 선택항목 -B 를 제공한다. 선택항목 -b 와 -B 들은 목적파일의 추론규칙의 가능한 위치와 실행, 탐색에 아무런 영향도 주지 않는다. .DEFAULT 가 아닌 다른 추론규칙의 탐색은 항상 진행된다.

SIGINT, SIGQUIT, SIGNUP, SIGTERM 와 같은 기발들은 목적파일이 특정의 이름 .PRECIOUS 에 종속되지 않으면 그 목적파일을 제거한다.

선택항목들

아래에서는 모든 선택항목들과 일부 특정의 이름들에 대하여 간단히 설명한다. 선택항목들은 임의의 순서로 나타날수 있다. 선택항목 -f 를 제외하고 모든 선택항목들은 개별적으로 혹은 한개의 -와 함께 규정될수 있다.

-b 낡은 제작파일을 위한 방식(방안 7)이다. 이 선택항목은 암시적으로 작용한다.

-B 낡은 제작파일을 위한 방식을 리용할수 없게 한다.

-d 오유검사방식이다. 파일들의 상세한 정보와 검사된 시간을 인쇄한다(이것은 대단히 큰 정보의 모임이고 make 지령자체를 정비검사한다.).

-e 환경변수들은 제작파일안의 인수들을 무시한다.

-f <제작파일>

제작파일로 참조되는 서술자파일이름을 규정한다. -는 표준입구장치장치를 표시한다. 제작파일의 내용은 그 파일들이 나타나면 체계규칙과 마크로를 무시한다. -f 와 제작파일사이에는 공백이 있어야 한다. 이 선택항목은 여러개를 사용할수 있으며(-f -는 제외) 규정된 순서로 처리된다.

-p 마크로정의와 목적파일서술의 완전한 모임을 표준출구장치장치에 기입한다.

-i 입장된 지령들에 의하여 귀환된 오유코드들을 무시한다. 특정의 목적파일이름 .IGNORE 가 제작파일에 나타난다고 하여도 이 방식으로 입장된다.

-k 지령이 0 이 아닌 상태를 귀환할 때 현재 구체례에 대한 작업은 끝내고 그 목적파일에 의존되지 않는 다른 가지들을 계속 작업한다. 이것은 -S 의 반대의 기능이다. 만일 -k 와 -S 가 둘다 규정되면 마지막에 규정된 선택항목이 리용된다.

-n 비실행방식을 지적한다. 지령들은 인쇄하지만 실행되지는 않는다. @로 시작한 행들도 인쇄된다. 그러나 문자렬 $(MAKE)와 ${MAKE}를 포함하는 행들과 지령에 앞붙이 +를 가지는 행들만 실행된다.

-q 질문을 규정한다. 지령 make 는 목적파일이 날자초과가 되였는가 안되였는가에 따라 0 혹은 0 이 아닌 상태코드를 귀환한

다. 목적파일들은 이 선택항목에 의하여 갱신되지 않는다.

-r 뒤붙이목록을 지우고 체계규칙을 리용하지 않는다.

-s 침묵방식을 지적한다. 지령들은 그것을 수행할 때 표준출구장치에 인쇄되지 않는다. 이 방식은 또한 특정의 목적파일이름인 **.**SILENT 가 제작파일에 나타나도 입장된다.

-S 목적파일의 날자초과를 가져 오는 지령을 실행할 때 오유가 나타난다면 종결한다. 이것은 기정값으로서 -k 의 반대기능이다. 만일 -k 와 -S 가 둘 다 규정되면 마지막으로 규정된 선택항목이 리용된다. 이것은 환경변수 MAKEFLAGS 에 있는 기발 k 의 존재를 무시하게 한다.

-t 목적파일을(시간초과로 간주하면서) 창조만 한다.

[macro_name = value]
한개이상의 지령행의 마크로정의를 규정할수 있다(마크로부분을 참고).

[names]
제작파일에 나타나는 목적파일의 이름으로서 없을수도 있고 여러개일수도 있다. 규정된 매 목적파일은 make 에 의하여 갱신된다. 만일 이름이 규정되지 않으면 make 는 추론규칙이 아닌 제작파일안의 첫 목적파일을 갱신한다.

환경

항상 무시되는 환경변수 SHELL 을 제외하고는 환경에서 정의된 모든 변수들은 make 에 의하여 읽어 지고 처리되며 마크로정의로서 넘겨 진다. make 는 자동적으로 SHELL 을 /usr/bin/sh 로 설정한다. 정의되지 않았거나 빈 문자렬로 정의된 변수들은 make 에 의하여 포함된다.

가능한 마크로정의는 본질(기정값), 현재의 환경, 제작파일, 지령행과 같은 순서로 읽어 진다. 이러한 처리순서때문에 제작파일에서 마크로기능은 환경변수를 무시한다. 선택항목 -e 는 환경이 제작파일에서 마크로기능을 무시하게 한다. 지령행마크로정의는 항상 임의의 다른 정의를 무시한다.

환경변수 MAKEFLAGS 는 그것이 지령행에서 정의된(-f, -p, -d 를 제외하고) 임의의 정해 진 선택항목을 포함한다는 가정우에서 make 에 의하여 처리된다. 환경변수 MAKEFLAGS 는 제작파일에서도 규정될수 있다.

XPG4 에서만은 다르다. 제작파일의 MAKEFLAGS 는 환경변수 MAKEFLAGS 로 교체된다. 지령행의 선택항목들은 MAKEFLAGS 환경변수보다 우선권을 가진다.

만일 MAKEFLAGS 가 어디에서도 정의되지 않으면 make 는 자체로 변수를 구성하고 지령행에서 규정된 선택항목들과 임의의 기정값선택항목들을 놓고 지령을 참조하여 그것을 통파시킨다. 그리하여 MAKEFLAGS 는 언제나 현재 입구되는 선택항목들을 포함한다. 이것은 귀납적으로 make 들에 대단히 유용한것이다. 선택항목 -n 이 규정될 때조차 문자렬 $(MAKE)와 ${MAKE}를 포함하는 지령행들은 수행된다. 때문에 무엇이 수행되고 있는가를 보려면 완전한 쏘프트웨어웨어체계에서 귀납적으로 make -n 을 수행할수 있다. 이리하여 -n 을 MAKEFLAGS 에 넣고 $(MAKE)와 ${MAKE}이 귀납적으로 발동하는것으로 전달될수 있기때문에 이것은 가능하다. 이것은 임의의 지령들이 실지적수행없이 쏘프트웨어웨어설계를 위한 제작파일을 모두 정비검사하는 한가지 방법인것이다.

규칙에서 매 지령들은 수행된 shell 에 주어 진다. 리용될 쉘은 쉘지령해석기이거나 SHELL 마크로로 제작파일에서 규정한 쉘이다. 마지막제작파일이 수행하는데 리용할 같은 쉘을 담보하기 위하여 다음과 같은 행을 리용한다.

SHELL=/usr/bin/sh 혹은 적당한 지령들을 제작파일의 마크로정의부분에서 놓을수 있다.

뒤붙이들

목적파일와 종속성이름들은 흔히 뒤붙이들을 가진다. 일정한 뒤붙이들에 대한 지식은 목적파일을 갱신하는데 적용되는 적당한 추론규칙들을 식별하는데 리용된다(추론규칙에 대한 절을 참고). 현재의 뒤붙이에 대한 기정값목록은 다음과 같다.

```
.o   .c   .c~  .C   .C~  .cxx   .cxx~  .cpp   .cpp~  .cc    .cc~
.y   .y~  .l   .l~  .s   .s~    .sh    .sh~
.h   .h~  .H   .H~  .p   .p~    .f     .f~    .r     .r~
```

이러한 뒤붙이들은 특수한 체계목적파일 .SUFFIXES 의 종속성으로서 정의된다. 이것은 make 에 의하여 자동적으로 진행된다.

추가적뒤붙이들은 .SUFFIXES 에 대한 종속성으로서 제작파일안에서 규정될수 있다. 이 추가적값들은 기정값들에 추가된다. 여러개의 뒤붙이목록들이 축적된다. 뒤붙이목록의 순서는 대단히 중요하다. 만일 사용자가 뒤붙이들의 순서를 변경시키려고 한다면 우선 빈 종속성목록으로 .SUFFIXES 을 정의하고(.SUFFIXES 의 현재값들을 지운다.) 그다음 요구되는 순서로 된 뒤붙이들로서 .SUFFIXES 를 다시 정의한다. 임의의 콤퓨터에서 리용되는 뒤붙이들의 목록은 다음과 같이 표시되여야 한다.

```
make -fp -2>/dev/null </dev/null
```

주어 진 mymakefile 이라는 제작파일안에 있는 정의에 부합되는 체계뒤붙이들의 목록은 다음과 같이 표시되여야 한다.

```
make -fp mymakefile -2>/dev/null</dev/null
```

추론규칙

.o 로 끝나는 일정한 목적파일 혹은 종속성이름들은 .c 와 .s 와 같은 추론가능한 종속성을 가진다. 만일 아무런 갱신지령들이 제작파일에서 나타나지 않고 추론가능한 종속성파일이 존재한다면 그 종속성파일은 목적파일을 갱신하도록 콤파일된다. 이 경우에 make 는 뒤붙이를 검사하고 뒤붙이들과 적당한 추론규칙을 검사함으로써 다른 파일들로부터 파일들을 구축하도록 하는 추론규칙을 가진다. 현재 표준적인 추론규칙들이 다음과 같이 정의되여 있다.

단일뒤붙이규칙들

```
.c    .c~
.C    .C~ .cxx .cxx~ .cpp .cpp~ .cc .cc~
.sh   .sh~
.p    .p~
.f    .f~
.r    .r~
```

2 중뒤붙이규칙들

```
.c.o .c~.o .c~.c c.a .c~.a
.C.o .C~.o .C~.C .C.a .C~.a
.cxx.o .cxx~.o .cxx~.cxx .cxx.a .cxx~.a
.cpp.o .cpp~.o .cpp~.cpp .cpp.a .cpp~.a
.cc.o .cc~.o .cc~.cc .cc.a .cc~.a
.s.o .s~.o .s~.a
.p.o .p~.o .p~.p .p.a .p~.a
.f.o .f~.o .f~.f .f.a .f~.a
.r.o .r~.o .r~.r .r.a .r~.a
.y.o .y~o .y.c .y~.c
.l.o .l~o .l.c
.h~.h .H~.H .hxx~.hxx .hpp~.hpp
.C.o .C~.o .C.a .C~.a
```

2 중뒤붙이규칙(.c.o)은 x.c 로부터 x.o 가 만들어 지는 방법을 정의한다. 단일뒤붙이규칙(.c)는 x.c 로부터 x 가 만들어 지는 방법을 정의한다. 첫번째 뒤붙이는 null 이다. 단일뒤붙이규칙은 한개로 된 원천파일로부터 목표파일을 만드는 경우에 효과적이다. 즉 쉘틀들과 단순한 C 프로그람들의 경우에 효과적이다.

우의 규칙들에서 ～은 SCCS 파일을 참조한다(sccsfile(4)를 참고). 이렇게 규칙 .c～.o 는 SCCS C 원천파일을 객체파일(.o)로 변환한다. SCCS 파일들의 s.은 앞붙이이기때문에 make 의 뒤붙이로 볼수 없다. 따라서 ～은 임의의 파일에 대

한 참조를 SCCS 파일에 대한 참조로 변경시키는 한가지 방법으로 된다.
뒤붙이 .c 를 가진 파일로부터 뒤붙이 .o 를 가진 파일을 창조하는 규칙은 아무런 종속성도 없으며 목적파일로서 .c.o 를 가진 구체례로서 규정된다. 목적파일과 관련되는 쉘지령들은 .c 파일로부터 .o 파일을 만드는 규칙을 정의한다. \이 하나도 없고 . 으로 시작하는 임의의 목적파일이름은 목적규칙대신에 추론규칙으로서 식별된다. 한개의 점을 가진 목적파일들은 단일뒤붙이추론규칙들이다. 두개의 점을 가진 목적파일들은 2 중뒤붙이추론규칙들이다. 사용자들은 제작파일안에서 추가적인 추론규칙들을 정의할수 있으며 표준적인 추론규칙들을 재정의하거나 취소할수 있다.

.c 파일을 .o 파일로 변경시키는 표준적인 추론규칙은 다음과 같다.

```
.c.o:
        $(cc) $ (CFLAGS) -c $<
```

파일 yacc 을 C 객체파일로 변경시키는 표준적인 추론규칙은 다음과 같다.

```
.y.o:
    $ (YACC) $ (YFLAGS) $<
    $ (cc) $ (CFLAGS) -c y.tab.c
    rm y.tab.c
mv y.tab.o $ @
```

일정한 마크로들은 임의의 결과지령들에서 선택적인수를 포함할수 있도록 표준적인 추론규칙에서 리용된다. 실례로 CFLAGS, LDFLAGS, YFLAGS 는 콤파일러선택항목으로서 cc(1), lex(1), yacc(1)에 각각 리용된다. LDFLAGS 는 련결자/적재자의 선택항목들을 지정하기 위하여 공통적으로 리용된다. 이러한 마크로들은 make 에 의하여 자동적으로 정의되지만 제작파일안에서 사용자에 의하여 재정의될수는 없다.

마크로 LIBS 는 습관적으로 콤파일의 련결과정에 어떤 특수한 서고의 포함순서를 규정하기 위하여 리용된다. 특정한 서고모임에 대한 포함순서를 특수하게 규정하려면 다음과 같은 .c 파일에 대한 이미 존재하는 단일뒤붙이규칙이 재정의된다.

```
$(CC) $ (CFLAGS) $< $ (LDFLAGS) -o $ @
```

이 규칙은 제작파일안에서 LIBS 정의를 정의하는것과 마찬가지로 다음과 같은것으로 재정의될수 있다.

```
$(CC) $ (CFLAGS) $< $(LDFLAGS) -o $ @ $ (LIBS)
```

추론규칙(@, <)에서 리용되는 몇가지 특수한 체계마크로들도 존재한다(체계마

크로부분을 참고).
목적파일이 명백한 종속성을 가지지 않거나 종속성이 명백한 규칙에 들어 맞는 목표파일을 가지지 않는다면 make 는 목표파일의 종속성의 뒤붙이(null 일수 있다.)에 들어 맞는 첫번째 추론규칙과 규칙의 다른 뒤붙이에 들어 맞는 파일을 탐색한다. .SUFFIXES 값들의 목록을 리용하여 거꾸로 다시 탐색할수 있기때문에 .SUFFIXES 이 정의되는 순서는 중요하다.

임의의 콤퓨터에서 make 에 의하여 콤파일되는 규칙들을 출구하려면 다음과 같이 입력해야 한다.

```
make -fp -2>/dev/null </dev/null
```

make 는 추론규칙 .c.o 를 정의하기때문에 일반해설부분에 있는 실례들은 다음과 같이 보다 단순하게 다시 쓸수 있다.

```
OBJS = a.o b.o
        pgm: $（OBJS）
                cc $（OBJS）-o pgm
        $(OBJS)：incl.h
```

서고들

만일 목적파일 혹은 종속성이름에 괄호가 포함되여 있으면 그것은 체계서고로 가정되며 괄호안의 문자렬은 그 서고안의 성분을 가리킨다고 간주한다. lib(file.o)와 $(LIB)（file.o)는 둘 다 file.o 를 포함하는 체계서고로 고찰된다(마크로 LIB 는 이미 정의되여 있다고 가정한다.). 식 $(LIB)(file1.o file2.o)는 틀린 식이다. 체계서고에 고유한 규칙들은 .xx.a 형식을 가진다. 여기서 xx 는 체계성분이 만들어 지는 뒤붙이이다. 다른 부차적인 성분들은 체계성분의 뒤붙이와 같은 xx 를 가질수 없다. 체계대면부의 가장 일반적인 리용은 다음과 같다. 여기서 원천파일을 모두 C 원천파일이라고 가정한다.

```
lib: lib(filel.o) lib(file2.o) lib(file3.o)
        @echo lib is now up-to-date
.c.a:
        $(CC) -c $(CFLAGS) $<
        ar rv $@ $*.o
        rm -f $*.o
```

<, @, *기호들에 대한 설명은 체계마크로부분을 참고하면 된다.
사실 우에서 현시된 .c.a 규칙은 make 에서 만들어 지며 이 실례에서 불필요한것이다. 이 규칙은 lib 의 매 종속성에 적용된다. 다음의 실례는 이것을 더 효과적

으로 수행하는 실례이다.

```
lib: lib(filel.o) lib(file2.o) lib(file3.o)
        $(CC) -c $(CFLAGS) $(?:.o=.c)
        ar rv lib $?
        rm $?
        @echo lib is now up-to-date
.c.a: ;
```

여기서는 마크로안에서의 치환이 리용되고 있다. 목록 $?는 시간초파인 원천파일들을 가지는 lib 안의 객체파일이름의 모임으로 정의된다. 치환렬은 .o 를 .c 로 변환한다(또한 규칙 .c.a 의 불가능성은 매개 목적파일을 하나씩 창조하고 관리한다는것을 주의하시오.). 이러한 특정의 구조는 체계서고를 합리적으로 유지관리하지만 체계서고에 아쎔블러어와 C 프로그람이 혼합되여 있는 경우에는 대단히 시끄러운 구조로 된다.

핵심부입력점은 입구점이름을 둘러 싼 2 중괄호에 의하여 lib ((entry_name))처럼 표시되지만 그밖의 경우에는 우에서 서술한것처럼 조종된다.

체계목적파일들

make 는 몇개의 특수한 목적파일들에 대한 지식을 가지고 있다. 이것들은 제작파일에서 효과를 가지도록 규정되여야 한다(.SUFFIXES 는 make 에 의하여 자동적으로 설정되지만 사용자에 의하여 변경될수는 없다.).

.DEFAULT

아무런 명백한 지령 혹은 관련되는 체계규칙이 없이도 파일이 만들어져야 한다면 목적파일이름 .DEFAULT 와 관련되는 지령들이 제작파일에서 정의되는 경우에 리용된다. .DEFAULT 는 그 어떤 명백한 종속성도 가지지 않는다.

.PRECIOUS

이 목적파일과 관련되는 종속성들은 QUIT, INTERRRUPT, TERMINATE, HANGUP 을 받을 때 제거되지 않는다.

.SILENT

선택항목 -s 와 동일한 효과를 나타낸다. 아무런 종속성 혹은 명백한 지령도 규정될 필요가 없다.

.IGONORE

선택항목 -i 와 동일한 효과를 나타낸다. 아무런 종속성 혹은 명백한 지령도 규정될 필요가 없다.

.SUFFIXES

.SUFFIXES 의 명백한 종속성이 체계뒤붙이들의 목록에 추가되며 추론규칙에 함께 리용된다. 만일 .SUFFIXES 가 아무런 종속성도 가지지 않으면 뒤붙이목록은 지워 진다. .SUFFIXES 과 관련된 지령은 하나도 없다.

체계마크로들

목적파일을 구축하는 규칙작성에 효과적으로 쓰이는 내부적인 5 개의 기본마크로가 존재한다. 이 마크로들의 의미를 정확히 정의하기 위하여 목적파일와 종속성에 대한 용어사용에서 주의할 점들이 있다. 명백하거나 명백치않은 임의의 규칙이 목적파일을 갱신하는데 적용되기전에 목적파일에 대한 매개 종속성에서 재귀적현상이 발생한다. 재귀적종속성은 목적파일 그자체이며 목적파일이 종속성을 가지지 않을 때까지 재귀적종속성을 가지거나 생성하게 된다. make 에 의하여 처리되는 모든 목적파일은 제작파일안에서 명백한 목적파일로 나타나지 않는다. 그것들의 일부는 제작파일안에서 명백한 종속성으로 나타나며 다른 일부는 make 가 목적파일들을 재귀적으로 갱신할 때 명백치 않은 종속성으로 나타난다. 실례로 다음과 같은 제작파일이 수행된다고 하자.

```
pgm:  a.o  b.o
        cc  a.o  b.o  -o  pgm
```

다음과 같은 목적파일렬이 발생한다.

--- pgm 두개의 종속성과 한개의 명백한 규칙이 뒤따르른다.

--- a.o (재귀적으로)명백치 않은 규칙 .c.o 에 들어 맞는 a.c 의 명백치 않은 종속성을 가진다.

--- a.c (재귀적으로)아무런 명백치 않은 종속성/규칙도 가지지 않는다. 이것은 재귀적순환을 정지시키며 파일 a.c 의 최종수정시간만을 단순히 귀환한다.

--- b.o (재귀적으로)명백치 않은 규칙 .c.o 에 들어 맞는 b.c 의 명백치 않은 종속성을 가진다.

--- b.c (재귀적으로)아무런 명백치 않은 종속성/규칙도 가지지 않는다. 이것은 재귀적순환을 정지시키며 파일 b.c 의 최종수정시간만을 단순히 귀환한다.

아래의 정의에서 목적파일이라는 말은 제작파일안에서 규정된 목적파일, make에 의하여 그자체에 대한 재귀적순환을 만들 때 목적파일로 되는 명백한 종속성 혹은 명백치 않은 종속성(목적파일의 뒤붙이를 들어 맞추는 추론규칙과 파일을 지적함으로써 생성된다.)을 의미한다. 종속성이라는 특수한 목적파일을 위하여 제작파일안에서 규정되는 명백한 종속성, 목적파일의 뒤붙이를 들어 맞추는 적당한 추론규칙과 대응하는 파일을 지적함으로써 생성되는 명백치 않은 종속성을 의미한다.

목적규칙들로서 특수한 목적파일이름에 대하여 사용자가 규정한 규칙들과 사용자에 의한 추론규칙들 그리고 목적파일이름들의 특수한 클라스에 대하여서는 make 가 규정한 규칙들을 고찰하는것이 효과적이다. 또한 목적파일이름들과 그것에 대응하는 종속성이름도명백하거나 명백치 않은 종속성들에 적용되는 추론규칙들(제작파일에서 정의되지 않는다.)을 기억하는것도 중요하다.

$@ 이 마크로는 완전한 현재의 목적파일이름 혹은 서고의 체계목적파일의 부분이름이다. 이것은 목적파일과 추론규칙 둘 다에 적용된다.

$% 이 마크로는 현재 목적파일이 서고이름(성분 .o) 혹은 서고이름((구체례 .o))형식의 체계성분일 때에만 효과가 있다. 이 경우에 $@는 서고이름을 위한것이고 $%는 성분 .o 혹은 구체례 .o 로 평가된다. 이것은 목적파일과 추론규칙 둘 다에 적용된다.

$? 이 마크로는 현재의 목적파일에 대한 날자초과종속성들의 목록이다. 본질적으로는 재구축되는 모듈들이다. 이것은 목적파일과 추론규칙 둘 다에 적용되지만 보통 목적파일규칙에만 리용된다. $?는 추론규칙에서 한개의 이름만 평가하지만 목적규칙에서는 한개이상의 이름을 평가할수 있다.

$< 추론규칙에서 $<는 목적파일의 뒤붙이를 들여 맞추는 명백치 않은 규칙에 대응하는 원천파일이름을 평가한다. 다시 말하면 이것은 목적파일에 대한 날자초과의 파일이다. .DEFAULT 규칙에서 마크로 $<는 현재 목적파일이름을 평가한다. $<는 추론규칙에서만 평가된다. 이렇게 .c.o 규칙에서 $<는 .c 파일을 평가하게 된다. .c 파일로부터 .o 파일을 최량화하는 실례는 다음과 같다.

```
.c.o:
        cc -c -O $* .c
or:
    .c.o:
        cc -c -O $<
```

$* 마크로 $*는 뒤붙이를 제거된 현재의 목적파일이름이다. 이것은 추론규칙에서만 평가된다.

이 5개의 마크로들은 여러가지 형식을 가질수 있다. D 혹은 F가 5개의 마크로중 임의의 마크로에 추가될 때 그것들의 의미는 D에 대해서는 등록부부분, F에 대해서는 파일부분으로 변경된다. $(@D)는 문자렬 s@의 등록부부분을 의미한다. 만일 등록부가 하나도 없다면 . /이 생성된다. 마크로 $?가 한개이상의 종속성이름을 포함할 때 $(?D)는 등록부이름부분들의 목록으로 전개되며 $(?F)는 파일이름부분들의 목록으로 전개된다.

우에서 서술한 체계마크로들외에 공통적으로 리용되는 다른 마크로들이 make에 의하여 정의된다. 이러한 마크로들은 표준적인 추론규칙으로 리용되며 선택항목 -p로써 표시할수 있다. 이러한 마크로들은 제작파일안의 목적규칙에서 리용될수 있으며 제작파일안에서 재정의될수도 있다.

$$@ $$@마크로는 종속성행에서만 의미를 가진다. 이러한 형식의 마크로들을 **동적인 종속성**(Dynamic Dependencies)이라고 부른다. 왜냐하면 이 마크로들이 실지로 종속성이 처리될 때에 평가되기때문이다. $$@는 $@가 지령행에서 진행하는것과 꼭같이 현재목적이름을 평가한다. 이 마크로는 한개의 원천파일만을 가지는 여러개의 실행파일을 구축하는데 효과적으로 리용된다. 실례로 다음과 같은 HP-UX 지령들은 꼭 같은 규칙을 리용하여 작성된다.

```
CMDS = cat echo cmp chown
$（CMDS）: $$@ .c
        $(CC) -O $? -o $@
```

만일 이 제작파일이 make cat echo cmp chown에 의하여 입장한다면 make는 cat가 목적파일인 경우 $$@로서 cat를 평가하면서 일반규칙을 리용하여 매개 목적파일들을 구축한다. 이때 목적파일들을 현시해준다.

동적인 종속성마크로는 F형식을 취할수 있다. $$(@F)는 $$@의 파일이름부분을 표시한다. 이것은 만일 목적파일이 경로이름을 포함한다면 효과적으로 리용된다(다음의 실례를 참고).

```
INCDIR = /usr/include
INCLUDES = $(INCDIR) /stdio.h \
        $(INCDIR) /pwd.h \
        $(INCDIR) /dir.h \
```

```
        $(INCDIR)/a.out.h
$(INCLUDES):$$(@F)
        cp $? $@
        chmod 0444 $@
```

특수마크로들

마크로 VPATH 는 make 가 :으로 구분된 종속성들을 위한 등록부목록을 탐색하도록 한다. VPATH = path1:path2…형식의 행들은 make 가 종속성을 위하여 현재등록부를 처음으로 찾도록 하며 만일 종속성이 발견되지 않으면 make 가 마크로 VPATH 에서 규정한 등록부들이 존재할 때까지 path1, path2, …를 계속 찾도록 한다.

외부적영향

환경변수들

LANG 은 설정되지 않았거나 비여 있는 국제화변수에 대한 기정값을 제공한다. 만일 LANG 이 설정되지 않았거나 비여 있으면 기정값 "C"가 리용된다(lang(5)를 참고). 임의의 국제화변수가 틀리게 설정되여 있으면 make 는 모든 국제화변수들이 "C"로 설정된것처럼 작용한다(environ(5)를 참고).

LC_ALL 이 비지 않은 문자렬값으로 설정되여 있으면 다른 모든 국제화변수들의 값들을 무시한다.

LC_CTYPE 는 본문의 단일 혹은 다중바이트문자에 대한 해석, 인쇄가능한 문자들의 모임, 정규식의 문자모임식에 들어 맞는 문자들을 결정한다.

LC_MESSAGES 는 표준오유장치에 출구되는 진단통보의 형식화된 내용과 표준출구장치에 출구되는 비형식화된 통보에 영향을 주는 지역을 결정한다.

NLSPATH 는 LC_MESSAGES 의 처리를 위한 통보분류의 지역을 결정한다.

PROJECTDIR 는 현재 등록부에서 찾지 못한 SCCS 파일들을 탐색하는데 리용되는 등록부를 제공한다. 다음과 같은 경우들에 SCCS 파일들에 대한 탐색은 식별된 등록부안에 있는 등록부 SCCS 안에서 진행된다. 만일 PROJECTDIR 의 값이 \으로 시작된다면 그것은 절대경로로 고찰된다. 그밖의 경우에는 보조등록부 src 혹은 source 를 위하여 검사된 사용자홈등록부로 고찰된다. 이러한 등록부가 있으면 리용되고 없으면 상대적경로이름으로 리용된다.

만일 PROJECTDIR 가 설정되지 않았거나 Null 값을 가지면 SCCS 파일들에 대한 탐색은 현재 등록부안의 등록부 SCCS 에서 진행된다.

PROJECTDIR 의 설정은 뒤부분의 성분이름으로서 SCCS 를 포함하는 모든 봉사프로그람파일들에도 영향을 미친다.

국제적인 코드모임보장

단일바이트 및 다중바이트문자코드모임이 보장된다.

귀환값

make 는 성과적으로 완성되면 령을 귀환하고 오유가 발생되면 0 보다 큰 값을 귀환한다. 선택항목 -q 가 규정된 경우에는 목적파일이 갱신되였을 때에는 0 을 귀환하고 갱신되지 않았을 때에는 0 보다 큰 값을 귀환한다.

실례

다음의 실례는 program.c 가 현재등록부에 있다면 제작파일이 없이 C 원천코드로부터 실행가능한 파일을 창조한다.

```
make program
```

다음의 실례는 한개이상의 규정된 제작파일들과 정의된 몇개의 지령행마크로들을 보여 주며 module1 에서 첫번째 목적파일을 갱신한다.

```
make -f modul1 -f modul2 RELEASE = 1.0 CFLAGS = -g
```

다음의 실례에는 현재 등록부안에 남아 있는 표준제작파일안에서 두개의 목적파일들을 갱신한다. 현재 있는 암시적제작파일에서 두개의 목적파일를 갱신한다.

```
make clobber prog
```

다음의 실례는 규정된 제작파일에 있는 목적파일 prog 를 갱신하며 환경변수들이 제작파일안에 있는 모든 공통변수들을 무시하도록 한다. 또한 체계뒤붙이목록을 지우고 체계규칙들을 무시하며 오유검사정보를 출구하도록 한다.

```
make -erd -f modul1 prog
```

경고

포함파일과 같은 임의의 파일의 접근, 수정, 최종변경시간은 make 처리과정에 변경될수 없다. 실례로 프로그람이 또 다른 포함파일에 종속되는 어떤 포함파일에 종속되고 이러한 파일들이 날자초과이라면 make 는 그것들이 수행될 때마다 갱신한다. 따라서 포함파일에 종속되는 갱신된 파일들에 대한 불필요한 재구축작업이 진행된다. 이 문제를 해결하기 위해서는 make 를 수행시키기전에 지령 touch 로 파일들을 갱신하겠는가를 물어 보는것이 좋다(touch(1)을 참고)(자기의 제작파일안에 지령 touch 를 포함시키는것은 일반적으로 좋지 않다. 왜냐하면 그것이 make 로 하여금 갱신할 필요가 없는 파일들까지도 갱신하도록 하기때문이다.).

일부 지령들은 정확치 않은 0 이 아닌 상태를 귀환한다. 이것을 피하려면 선택항목 -i 를 리용하면 된다.

= : @ $ 들을 포함하는 파일이름들은 작업을 할수 없다.

지령 cd 와 같이 쉘에 의하여 직접 실행되는 체계지령들은 make 에서 여러 행에 쓸수 없다.

(lib(file1.o file2.o file3.o))형식은 틀린 형식이다.

file.o 로부터 lib(file.o)를 만들수 없다.

마크로 $(a : .o = .c ~)는 작업할수 없다.

확장된 목적행들은 새 행문자까지 포함하여 16384 개이상의 문자를 포함할수 없다.

make 가 null 로 평가되는 " 인수를 가지고 쉘스크립트에서 입장된다면 make 는 실패한다.

종속성

NFS 경고

다른 NFS 봉사기우에 있는 파일들의 수정시간을 비교할 때 make 는 봉사기의 시계가 동기화되지 않으면 make 는 예견할수 없는 행동을 한다.

파일들

[Mm] akefile
s. [Mm] akefile
SCCS/s . [Mm] akefile

참고

cc(1), cd(1), lex(1), mkmf(1), sh(1), yacc(1), environ(5), lang(5), regexp(5)

A Nutshell Handbook, Managing Projects With Make by Steve Talbot, Second Edition, O ' Reilly & Associatas, Inc., 1986.

표준일치

make : SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

제２２장. C 프로그람작성의 기초

이 장에서는 C 와 C++프로그람작성법에 대한 많은 기초자료들을 소개하기로 한다. 여기서 이 자료들은 C 언어로 소개되여 있다. 프로그람작성부분에서는 먼저 소개를 어떻게 하는가에 대하여 보기로 한다.

C 프로그람들은 함수단위로 제시된다. 한개의 프로그람은 함수들에로 이행하면서 론리적인 단위로 수행된다. 함수들은 보통 서로 다른 함수들과 명령문들로 만들어 진다. 명령문들은 과제를 수행하는 코드로 된 행들이다.

C 프로그람은 main( )이라고 부르는 함수를 가지고 있다. main 함수는 모든 C 와 C++프로그람의 시작함수이다. 그림 22-1 에서 보는바와 같이 main( )부분은 프로그람본체이다.

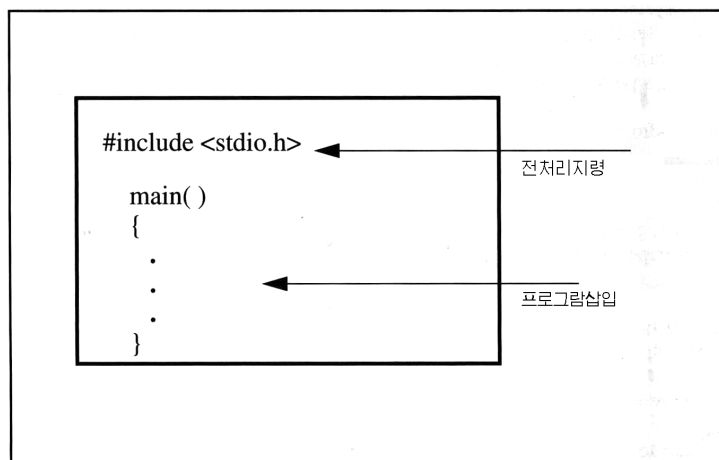



그림 22-1. C 프로그람 main( )

프로그람본체는 명령들과 함수들로 이루어 져 있다. **전처리지령**(Preprocessor Directive)들은 함수 main( )밖의 우에 기입된다.

하나의 완성된 C 프로그람은 전처리지령 stdio.h 와 프로그람본체인 main( )함수로 이루어 진다. 프로그람은 확장자 ".c"를 가진 파일로 되여야 한다.

그림 22-2 에 있는 C 프로그람인 TestAverag.c 는 완성된 C 프로그람이다.

이 프로그람은 산수연산을 진행하고 결과를 출력한다. 이 프로그람은 일단 정확히 **콤파일**(Compile)되면 실행될수 있다. 이 프로그람은 시험점수를 받아 그 시험점수들을 더하고 3 으로 나누어 평균값을 계산한다. 그리고 마지막에 성적을 출력한다. 시험점수와 평균값은 한개 문장으로 되여 있으며 화면에 출력되여 있다.

프로그람에서 임의의 설명문들은 /*와 */기호사이에 놓인다. 설명문들은 콤파일러에 의하여 무시된다. 설명문을 포함시키는것은 프로그람을 작성하는데서 좋은 습관으로 된다.

```
/* 파일이름  TestAverage.c
이 프로그람은 하나의 C서고에 보관된 인쇄기능를 리용하여
결과들을 인쇄하며 세개 시험점수의 평균을 계산한다.              ←—— 설명문

#include <stdio.h>                                              ←—— 전처리지령

main()                                                          ←—— 기본함수
{                                                               ←—— 블로크시작
   int grade1, grade2, grade3, total_score;                     ←—— 변수소개
   float_average;

   grade1=96;                          /*assign grades*/
   grade2=89;
   grade3=99;

   total_score = grade1 + grade2 + grade3;   /*add grades*/

   average = total_score/3;            /*find average grade*/

   printf("For test scores %d, %d and %d the final grade is %f",
          grade1, grade2, grade3, average);                     ←—— 화면에 출구하는
                                                                     명령문
   /* programs and functions should end with a return statement*/

   return;
}                                                               ←—— 블로크끝
```

그림 22-2. C 프로그람 TestAverag.c

처음에 전처리지령이 있고 다음에 main( )함수가 놓인다. 왼쪽 대괄호({ }는 함수의 본체인 코드블로크의 시작을 나타낸다. 차례로 쓴 변수들은 자료형을 지적하고 값들을 보관한다. 보충적인 변수들은 합을 계산하고 평균을 계산하기 위하여 소개된다. 매개 점수들을 모두 더한 다음 평균값이 계산된다. 함수 printf( )는 표준 C 서고의 부분이며 화면에 출력하는 함수이다. 함수의 본체에 있는 원천코드의 마지막행은 귀환명령이다. 귀환명령은 호출된데 따라 함수에로 값을 돌려 보내게 한다. main( )함수가 값을 귀환하면 그 값은 UNIX 쉘에 보내진다. 모든 함수들은 귀환명령으로 끝나야 한다. 오른쪽 대괄호 ")"는 코드블로크의 끝을 정의한다.

프로그람의 출력은 단순하다. 시험점수는 마지막변수에 들어 간다. 이 마지막변수는 세개 시험점수들의 평균값이다.

C 프로그람의 형식

C 는 자유형언어이다. 즉 프로그람을 작성하는 명령문들은 임의의 행, 임의의 렬에서 시작할수 있다. 공백행들이 있을수 있는데 이것들은 무시된다. C 언어는 한개 행에 한개 명령문이 있을것을 요구하지 않는다. 모든 명령문의 끝에는 반두점(;)이 있어야 하

며 명령문들은 련이어 놓일수 있다.

프로그람작성에서 좋은 습관

C 프로그람을 쉽게 리해할수 있게 하기 위한 방법에는 여러가지가 있다.

우선 프로그람에 대한 설명을 하는 설명문을 리용하는것이 좋다. 원천파일을 서술할 때에는 설명문을 쓰는것이 좋다. 설명문은 파일에서 함수들이 어떤 처리를 하는가를 설명하여 준다. 실례로 일부 자료에는 원천파일의 이름, 저자, 창조된 날자, 프로그람이 하는 처리들이 포함시킬수 있다.

다음으로 변수의 이름은 자료의 내용에 맞게 쓰는것이 좋다. 함수이름들은 그것들의 목적과 동작내용을 반영하여 서술할수 있다.

C 명령문들은 개별적인 행에 기입한다.

개별적인 행들과 명령문들은 리해하기 편리하게 빈 공간을 배치하는데 이것을 잘 리용해야 한다. 함수와 코드블로크를 표시하는데 빈 공간을 배치하면 프로그람을 보는 사람이 편리하게 할수 있게 한다. 원천코드는 대괄호안에 들어 간다.

```
function_1( )
{
(body of function is indented)
}
```

프로그람전반에 걸쳐 설명문을 사용하면 좋다.

설명문

설명문을 사용하면 프로그람을 리해하기가 쉽다. 설명문은 /*로 시작되고 */으로 끝난다. 그것들사이에 있는 모든것은 설명문에 해당되며 콤파일러에 의해 무시된다.

설명문은 코드의 한개 행을 차지할수도 있고 여러개 행을 차지할수도 있다. 설명문은 한개 행에서 명령문앞에 삽입될수도 있고 끝에 첨가될수도 있다. 사용자는 프로그람을 작성하면서 필요할 때 설명문을 쓰는것이 좋다.

일부 프로그람작성자들은 원천코드가 완성된후에 설명문을 쓰려고 할수 있는데 이것은 후에 시간이 모자라면 쓰지 못할수도 있다. 또한 후에 써넣는 설명문은 정확하지 못할수도 있고 명백하지 않을수도 있다.

대문자와 소문자

대문자와 소문자들은 다른 프로그람언어보다도 C 언어에서 더 의의를 가진다. 사용자들은 TestAverage.c 에 있는 프로그람의 대부분이 소문자라는것을 알수 있다. C 언어에서는 지령도 대문자로 쓰지 않는다. 여기서 전처리지령은 례외로 된다. 이에 대해서는 이 장의 마지막에 서술된다.

C 언어의 요점

여기서는 기초개념부터 시작하여 여러가지 내용들이 서술된다.

표준서고

표준서고는 콤파일러와 함께 있으며 여기에는 함수의 표준번호들이 있다. 이 함수들은 미리 작성되여 있다. 이 표준함수들은 건반에서 자료를 읽기도 하고 파일로부터 통보를 읽기도 하며 그리고 화면에 통보를 표시하며 일부 수학적루틴에 대한 표준기능도 가지고 있다.

표준서고가 제공해 주는 모든 함수들을 설명하자면 한개장으로도 될것이다. 여기서는 가장 일반적인 함수만을 제시해 준다. 또한 함수들을 리용하는데서 필요한 머리부파일들의 목록을 개괄한다.

표준서고를 리용하기 위해서는 원천파일에 전처리지령 #include 로 표준머리부파일들을 포함하여야 한다.

상수

이미 앞에서는 변수에 대한 개념을 고찰하였다. 상수도 콤퓨터기억에 보관되는 값이기는 하지만 이것은 값이 변하지 않는다.

변수에 상수를 대입할수 있다. 아래에서는 상수를 변수에 어떻게 대입하는가를 보여주었다.

```
pens=10;
```

이 실례는 상수 10 을 변수 pens 에 어떻게 대입하는가를 보여 준다. 이 명령을 수행시키면 10 이라는 값이 변수 pens 에 들어 간다.

상수의 몇가지 류형을 보자. 옹근수상수는 0 부터 9 까지의 수자들을 리용하여 만들수 있다. 문자상수는 하나의 웃반점('z')안에 쓴다. 류점상수는 소수점과 0 부터 9 까지의 수자들을 리용한다.

기호상수

기호상수들은 상수의 실제적인 값대신 프로그람에서 리용될수 있는 이름 혹은 식별자들이다. 프로그람이 콤파일되면 전처리지령은 기호상수지적자에 대한 원천코드를 찾아 그 이름 혹은 식별자에 실제적인 상수값을 준다.

정의지령(Define Directive)은 기호상수를 정의하는데 리용된다. 정의지령은 전처리지령이 이름 또는 식별자에 값을 준다는것을 의미한다. 전처리지령은 기호(#)로 시작된 명

령문이다. 정의지령은 #define 과 그뒤에 있는 식별자와 상수로 이루어 져 있다. 다음의 지령은 pi에 값을 지적하는 기호상수를 정의한다.

```
#define PI 3.14
```

기호상수는 옹근수 혹은 류점수와 같은 상수가 프로그람전반에서 리용되는 경우에 편리하며 이것은 다시 정의될수도 있다. 실례로 마일에 해당한 값을 계산하는 프로그람을 들수 있다. 마일당 지불값은 때때로 변할수 있다. 지불값은 .32 에서 .33 까지 변할수 있으며 프로그람작성자는 프로그람에서 .32 가 되는 때를 찾으면 그것을 .33 으로 고치도록 하려고 한다.

이때 좋은 방법은 정의지령을 리용하여 지적자를 상수로 지적하는 방법이다. 지적자는

```
#define PER_MILE .32
```

로서 한번 나타나며 이 지적자는 원천파일끝 혹은 머리부파일의 우에 제시될수 있다. 프로그람작성자는 프로그람전반에서 실제적인 상수값 .32 대신 지적자 PER_MILE 을 리용한다. 지불값이 변하면 프로그람작성자는 정의지적자만을 바꾸면 된다.

탈퇴렬

일부 기호들은 **탈퇴렬**(Escape Sequences)들로 리용될수 있다. 문자상수로서 /새행/ 혹은 /태브/로 표현된것은 직접 수행될수 없다. 입력시킬 때 그것들은 원천파일을 범위내에서 새행 혹은 태브로 이행한다.

이 조종문자들은 줄바꾸기의 수단으로서 상수처럼 입력된다. 탈퇴렬들은 조종을 간접적으로 진행한다. 탈퇴렬들은 언제나 거꿀빗선(\)으로 시작된다.

표 22-1 에서는 탈퇴렬들을 보여 주고 있다.

표 22-1 탈퇴렬

조종문자	탈퇴렬
새행	\n
수평뛰여넘기	\t
수직뛰여넘기	\v
뒤공백	\b
돌아오기	\r
형태끝	\f
경보신호	\a
거꿀빗선	\\
웃두점	\"
의문부호	\?

자료형

C 언어에는 4 가지 기본적인 자료형들 즉 옹근수, 류점수, 배정확도류점수, 기호자료형들이 있다. 다른 자료형들은 여기서 론의하지 않기로 한다. C 언어에서는 어떤 형태의 자료형이 리용되는가를 알아 둘 필요가 있다. 자료형태는 리용되는 변수의 자료형과 함수로부터 귀환되는 자료형을 정의하기 위하여 리용된다. 자료형과 함수들에 대하여 더 보려면 함수들에 대한 부분을 보기 바란다.

아래의 실레에서 제시된 프로그람에서는 변수 road_exit 가 옹근수형만을 포함하도록 정의하였다. 변수정의는 함수의 본체안에서 프로그람작성명령문앞에 놓인다.

```
void directions_home( )
{
    int road_exit;      /*변수자료형정의*/
                        /*프로그람작성명령문들의 삽입*/
}
```

프로그람이 어떤 자료형을 처리하는가 하는것을 보면 자료취급방식과 어떤 연산이 정확한가 혹은 부정확한가 하는것을 알수 있다. 자료형은 그것이 함수의 본체에서 리용되기에 앞서 변수들로 정의 된다.

변수정의형태는 자료형, 변수이름을 쓰고 그뒤에 반두점을 쓴다.

```
data type variable_name;
```

한 행우에 같은 자료형태의 변수들을 여러개 정의하려면 변수이름들을 반점으로 구분하여야 한다.

```
data type variable_name1, variable_name2;
```

자료형을 정의할 때 변수에 상수를 값주기명령으로 줄수 있다. 이것은 변수를 초기화한다든가 변수에 초기값을 준다든가 할 때 리용된다.

```
data type variable_name 1=constant;
```

변수들이 정의된 다음에 그 변수에 값을 주면 초기값이 정의된다.

```
variable_name1=constant;
```

매개 자료형은 해당한 부분에서 서술하게 된다.

옹근수

옹근수는 소수점을 가지지 않는 임의의 정수 또는 부수이다. 옹근수는 모두 수자들로만 구성되게 된다. 변수형 옹근수는 C 프로그람에서 int 로서 정의된다.

```
int grade1;
```

옹근수들은 부호 + 혹은 −를 가질수 있다. 옹근수에는 반점, 소수점 또는 "$"와 같은 특수기호들이 포함될수 없다. 옹근수의 실례는 다음과 같다.

```
10 -15    +999   256    -26112
```

매개 콤퓨터체계는 여러가지 옹근수형을 리용할수 있는 수들의 한계에 따르는 제한을 가지고 있다. 이것은 매개 자료형에 대하여 콤퓨터가 보장해 주는 기억방법에 관계된다. 매개 바이트는 비트로 이루어 진다. 비트는 콤퓨터가 처리할수 있는 가장 작은 단위이며 0 혹은 1 의 값을 가진다. 한 바이트는 8 개 비트로 이루어 져 있다. 한 바이트는 각각 한개의 문자, 수자 또는 기호를 표현한다.

콤퓨터체계에서의 기억배치를 알려면 콤퓨터참고안내서를 보면 된다.

일부 기억배치에 대한 내용을 표 22−2 에 제시하였다.

표 22−2　　　공통적인 기억할당

할당된 기억	최대옹근수값	최소옹근수값
1 byte	127	-128
2 byte	32767	-32768
4 byte	2147483647	-2147483648

최대로 표시할수 있는 옹근수범위가 너무 작을 때에는 부호 없는 옹근수를 리용할수 있다.

```
unsigned int my bank_acct;
```

부호가 없는 형으로 소개된 변수는 정수만을 취한다. 표 22−3 에서는 부호 없는 옹근수를 보여 준다.

표 22−3　　부호 없는 옹근수의 기억할당

할당된 기억	최대옹근수값	최소옹근수값
1 byte	255	0
2 byte	65535	0
4 byte	4294967295	0

류점수

류점수들은 소수점을 포함하는 수이다. C 언어에서 float 형으로 정의된다.

```
float acct_total;
```

아래에서는 류점수에 대한 실례를 보여 주고 있다.

```
+16.776   5.    -6.2    3244.25 0.0    0.222   11.0
```

류점수에서는 옹근수에서와 마찬가지로 반점과 특수기호가 허용되지않는다. 류점수형의 변수에 보관되는 자리수는 콤퓨터체계에 의존한다. 류점변수들은 매우 크거나 작은 수들을 나타낼수 있으나 옹근수에서처럼 제한을 가지고 있다.

류점수형태의 변수가 가질수 있는 최소값은 1.0e-37 이며 최대값은 1.0e38 이다.

배정확도수

배정확도수는 류점수형태와 비슷하다. 배정확도수는 double 로 정의된다.

```
double  acct_total;
```

류점수형은 정확도가 높지 않을 때 리용된다. 배정확도수형태로 정의된 변수들은 류점수형태와 류사하지만 유효수자가 두배이다. 보관되는 정확한 자리수는 콤퓨터체계에 관계된다. 류점형의 변수와 마찬가지로 배정확도형변수들은 반점과 특수기호를 포함하지 않는다.

배정확도수의 최소, 최대값은 콤퓨터체계에 의존하지 않는다. 그것들은 류점형변수들에서 고찰한 최대, 최소한계까지 확장된다.

지수표기법

류점수와 배정확도수들은 지수로 표시할수 있다. 지수는 일반적으로 아주 큰 수를 표시하는데도 리용하고 아주 작은 수들을 표시하는데도 리용한다.

여기에서 문자 e 는 제곱을 표시하기 위하여 리용한다. 결수다음에 오는 문자 e 는 10 을 대신하는데 제곱수는 소수점이 10 진수의 표준형태로 얻어 질 때 이동되여야 할 자리수를 지적한다. e 다음에 놓이는 수가 정수이면 소수점은 오른쪽으로 지적된 자리까지 이동된다. e 다음에 놓이는 수가 부수이면 소수점은 왼쪽으로 이동한다.

실례로 e5 는 소수점을 오른쪽으로 5 자리만큼 옮겨야 한다는것을 지적한다. 그러므로 수자 5.897e5 는 589700 로 된다. 수 1.664e-3 에서 e-3 은 소수점을 왼쪽으로 세자리만큼 옮겨야 한다는것을 의미하며 이때 수자는 .001664 로 된다.

기호

C 언어에서 네번째 자료형태는 기호형이다. 기호는 자모기호, 10 개의 0 부터 9 까지의 수자와 ! * $, + @와 같은 특수기호로 이루어 진다. 기호는 ' '에 포함된 임의의 한개의 문자, 수자 또는 특수기호로 이루어 져 있다. 실례는 다음과 같다.

```
'A'   '9'   ' '   '!'   '+'   'c'
```

기호형변수는 다음과 같이 정의된다.

```
char  letter_grade;
```

단순기호들을 문자기호렬과 혼돈하지 말아야 한다.

실례로 문자들 'B', 'e', 'a', 'c', 'h'는 기호렬 'Beach'와 다르다. 'B', 'e', 'a', 'c', 'h'와 같이 표시된 매개 문자들은 우리가 자동적으로 Beach 라는 단어로 읽을수 있지만 콤퓨터는 개별적인 문자로 보게 된다. 'B'는 기억기의 첫 위치에 보관되며 'e'는 다음 위치 등으로 보관된다. 콤퓨터는 매개 문자를 그자체의 론리적단위로 본다.

사람들이 Beach 라는 단어로 읽기때문에 C 언어는 우리에게 문자들을 하나의 "단어"나 기호렬로 합치는 방식을 제공한다. 기호렬은 문자렬, 0 부터 9 까지의 수자 그리고 웃두점 (" ")에 포함된 특수문자로 이루어 져 있다. 기호렬 "Beach"에 있는 문자들은 콤퓨터안에서 린접되여 보관되거나 하나하나 련달아 있을수 있다. C 언어에서는 매개 문자를 개별적으로 처리하기가 힘들기때문에 기호렬함수의 표준설정과 리용을 설정해줌으로써 이 리용은 문자들을 쉽게 처리하게 해준다. 기호렬에 대해서는 앞으로 고찰하게 된다.

Void

void 자료형은 크기와 값범위를 가지지 않는 일반적인 자료형이다. void 값은 존재하지 않는다. 즉 값은 void 자료형변수에 보관될수 없다. 사용자가 void 형변수에 값을 주려고 한다면 콤파일러는 오유통보를 내보낸다. void 는 함수가 귀환값을 보내지 않는다는것을 콤파일러에 통보한다.

```
main( )
{
        void  function_1( );
}
```

이 자료는 "함수들"부분에서 자세히 론의하게 된다.

산 수 식

옹근수들, 류점수 및 배정확도수들은 더하기(+), 덜기(-), 곱하기(*), 나누기(/)연산을 할수 있다. 산수식을 처리할 때에는 자료형에 따르는 연산인수들을 갈라 놓는다. 산수연산은 문자자료형에 대해서도 할수 있다.

매개의 산수연산자 (+, -, *, /)는 2 항연산으로서 두개의 연산인수를 요구한다. 이 연산자들은 산수연산이 수행되도록 산수연산자의 매 부분에 어떤 객체가 놓일것을 요구한다. 연산인수들은 산수연산자의 왼쪽과 오른쪽에 놓인다. 연산자(+, -, *, /)는 먼저 왼쪽 연산인수를 계산하며 그 다음에 오른쪽 연산인수를 계산한다. 그림 22-3 에서는 산수식의 실례를 보여 주었다.

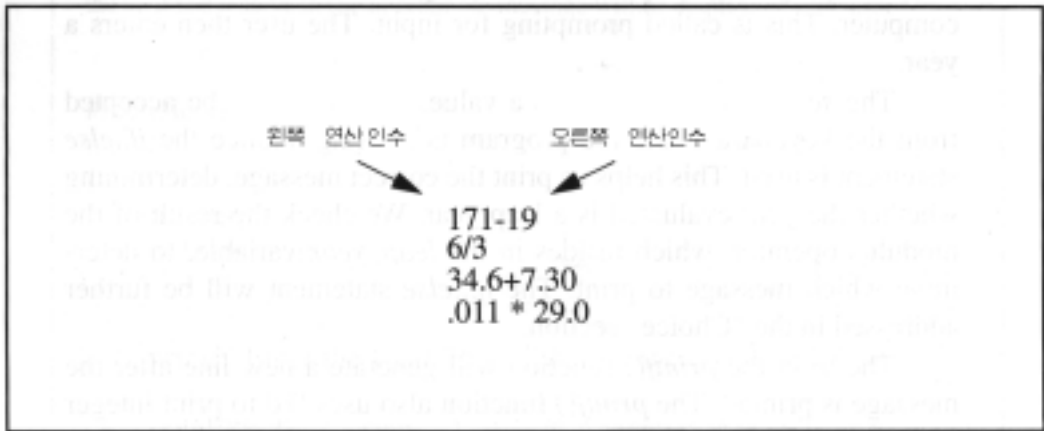



그림 22-3. 산수연산자들

연산자의 옆에 공백이 놓일수도 있다.

산수식의 자료형태에 따라 결과는 두가지 방법으로 계산한다. 그 방법을 보면 다음과 같다.

① 모든 연산인수가 옹근수(int)형자료이면 결과값은 옹근수이다.

② 어느 연산인수가 류점수(float)형자료이거나 배정확도(double)자료형이면 결과는 배정확도수이다.

다섯번째 산수연산자인 나머지(%)연산자도 2 항연산자이다. 이 연산자는 나누기를 한 다음 나머지를 첫 연산인수에 남겨 둔다. 실례로 7%4 는 3 이다. 왜냐하면 7 을 4 로 나누면 나머지가 3 이고 상이 1 로 되기때문이다. 상을 무시하므로 나머지가 결과값으로 된다.

이 연산자는 옹근수에서만 쓸수 있다. 임의의 다른 자료형변수가 리용되면 콤파일러는 오유를 통보한다.

그림 22-4 에서는 윤년을 계산하는데 나머지연산자를 리용한 프로그람을 주었다.

프로그람은 콤퓨터화면에 "Enter the year:"을 표시한다. 이 표시를 **입력재촉문**(Prompting for Input)라고 부른다. 그 다음에 사용자는 년도를 입력시킨다.

함수 scanf( )는 프로그람이 실행되고 있는동안 건반에서 입력된 년도의 값을 받는다. if…else 명령문은 지적된 년도가 윤년이 아닌가를 검열하는데 리용된다. 프로그람에서는 나머지연산자의 결과를 검사하고 그 결과를 leap_year 로서 화면에 출력한다. if…else 명령문은 뒤부분에서 자세히 설명하게 된다.

함수 printf( )에 있는 \n 은 통보가 나타난후에 행바꾸기를 진행한다. 또한 함수 printf( )는 옹근수변수들을 표시하는데서는 %d 를 리용하며 류점변수를 표시하는데서는 %f 를, 기호변수를 표시하는데서는 % c 를 리용한다.

```
/* 프로그람 leapyear.c는 사용자들이 년도를 입구하면
   어느 해가 윤년인가를 계산한다.
   필요한 통보가 화면에 나타난다. */

#include <stdio.h>
main()
{
      int year;
      int leap_year;


      printf("Enter the year: ");         /* 사용자에게 재촉한다. */

      scanf("%d", &year);                 /* 사용자로부터 년을 접수한다. */

      leap_year = year % 4;               /* 나머지 연산자를 사용한다. */

      if (leap_year == 0)                 /* 윤년이다. */
              printf("%d is a leap year.\n", year)

      else                                /* 윤년이 아니다. */
              printf("%d is not a leap year.\n", year)

      return;
}
```

그림 22−4. C 프로그람 leapyear.c

부호산수연산자는 하나의 연산인수만을 가진다. 미뉴스(-)연산자를 적용한 결과의 연산값은 부수가 된다.

실례로 부호 미뉴스는 정수를 부수로, 부수를 정수로 만든다. 더하기(+)연산자를 적용하는것은 연산인수의 값에 영향을 미치지 않는다.

아래에 이것을 설명하는 C코드의 부분을 제시하였다.

```
count=10;
a_difference=-count;          /*a_difference는 현재 -10을 포함한다.
                                왜냐하면 계산값이 미뉴스연산자에 의해 변환되
                                였기때문이다. */
```

```
no_difference=+a_difference;        /*no_difference는 현재 -10을 포함한다.
                                       왜냐하면 뿔류스연산자가 a_difference의 값에 영
                                       향을 주지 않기때문이다. */

no_difference=-a_difference;        /*no_difference는 현재 10을 포함한다.
                                       왜냐하면 미뉴스연산자가 a_difference 값을 변화
                                       시키기때문이다. */
```

증가연산자와 감소연산자

증가(++)연산자는 한개 연산인수의 값에 1을 더하며 감소(--)연산자는 1을 던다. 이것은 부호연산자이다. 왜냐하면 한개 연산인수를 요구하기때문이다. 사용자는 이 연산자들의 값에서 1을 덜거나 더하기 위한 간단한 부호로 생각할수 있다.

실례로 더하기연산자를 리용하여 어떤 수에 1만큼 더하려면 다음과 같이 할수 있다.

```
counter=counter+1;
```

증가연산자를 리용하면 같은 계산을 더 쉽게 할수 있다.

```
counter ++;
```

두 명령은 같은 일을 수행한다. 즉 변수 counter의 값에 1을 더한다는것이다. 옹근수나 류점수와 배정확도수와 같은 임의의 수자연산인수에 대하여 증가연산자 및 감소연산자를 리용할수 있다.

증가 및 감소연산자에는 두가지 종류가 있다. 연산자가 연산인수앞에 놓일 때 **앞붙이증가**(++counter) 또는 **앞붙이감소**(--counter)라고 부르며 연산자가 연산인수뒤에 놓일때에는 **뒤붙이증가**(counter++) 또는 **뒤붙이감소**(counter--)라고 부른다. 앞붙이연산자는 그 연산이 리용되기전에 연산인수를 변화시키며 뒤붙이연산자는 그 연산이 리용된 다음에 연산인수를 변화시킨다.

이 상태를 설명하기 위한 C코드의 토막을 아래에 제시한다.

```
int bottles=3;          /*변수를 정의하고 초기값을 준다. */
int cans=2;
bottles++;              /*bottles는 현재 4이다. */
cans=bottles++;         /*cans는 bottles의 4로 값주기되였고
                          bottles 는 5 로 증가되였으며 따라서 cans 는 현재 4,
                          bottles는 5이다. */
botttles++;             /*bottles는 현재 6이다. */
cans=--bottles;         /*bottles는 먼저 5로 감소되고 그다음
                          5를 cans에 준다. 따라서 bottles는 현재 5, cans는 5
                          이다. */
```

대입연산자

대입연산자는 값들을 변수에 준다. 가장 기본적인 대입연산자는 같기기호(=)이다. 값을 주려면 변수는 대입연산자의 왼쪽에, 값은 오른쪽에 주어야 한다.

```
dalmatians=101;
```

변수 dalmatians 는 101 을 포함한다.

C 언어에는 간단한 표시를 위하여 =뿐아니라 다른 연산과 결합한 다른 대입연산자도 가지고 있다.

명령문은 표준적으로 다음과 같이 쓴다.

```
biscuits = biscuits*2;
```

와 같이 씌여 진 명령문을

```
biscnuts*=2;
```

와 같은 다중연산자와 결합한 대입연산자로 쓸수 있다.

더 복잡한 식을 보면

```
biscuits / = kibbles + treats;
```

와 같은데 이것은 다음의 명령문과 일치한다.

```
biscuits = biscnits / (kibbles + treats);
```

대입연산자를 포함한 결합연산자는 다음과 같다.

```
+ =     - =     * =     / =     % =
```

형변환

C 언어는 형변환을 할수 있게 한다. 형변환은 형이 정의된후에 변수나 상수, 식의 자료형안에서 일시적인 변환이며 항상 연산들사이에 여러가지 자료형을 비교하는데 리용된다. 자료형들의 변환은 자동적으로 콤파일러규칙에 따라 콤파일러에서 진행되거나 프로그람작성자가 변환연산자를 가지고 진행한다.

식은 변수, 상수 또는 식 등과 여러가지 자료형으로 이루어 지기때문에 콤파일러는 무슨 자료인가를 확증해야 한다. 실례로 식이 옹근수나 류점수로 이루어 지면 모든 수자들은 류점수로 된다.

식이 옹근수와 류점수 그리고 배정확도수를 포함하면 모든 수는 배정확도수로 된다. 왼쪽정합이 진행되면 큰 수의 부분이 더 작은 자료형으로 맞추어 진다.

대입연산자의 왼쪽연산인수가 오른쪽연산인수보다 더 작은 자료형일 때에는 왼쪽정합이 진행된다. 실례로 식이 류점수로 이루어 지면 결과값은 류점수로 된다. 이 수가 옹근수변수에 보관되면 왼쪽정합이 일어 난다. 이때 류점수의 정확도는 낮아 진다.

```
float data1=3.0, data2=2.0;
int results;
results=data1/data2;
```

대입연산자의 오른쪽 연산인수인 식 data1/data2 는 류점수 1.5 로 계산된다. 이 수는 왼쪽 연산인수에 계산된 값을 주므로 결과는 옹근수변수로 나타난다. 실례에서 이 옹근수변수의 결과는 1 로 된다. 왜냐하면 류점수의 정확성이 낮아 지기때문이다.

C 언어는 변화연산자를 제공하는데 이것은 프로그람작성자가 처음에 소개한 다음 식, 변수, 상수를 다른 자료형으로 변환한다. 이런 조작은 마지막까지 원래의 변수나 그 내용에 영향을 주지 않으며 림시적으로 요구되는 자료형태로 변수들을 처리한다.

변화연산자는 (data type)의 형태를 가지며 자료형태는 괄호안에 써야 한다. 변화연산자는 그뒤에 있는 상수, 변수, 식을 변환연산자의 자료형태로 변환한다. 변화형태는 다음과 같다.

(data type) variable

혹은

(data type) constant

혹은

(data type) (expression)

아래의 실례는 류점수가 있는 식에서 단순변수의 계산에 대하여 보여 준다.

```
float results, data_3=5.5;
int data_1=3, data_2=2;

results=((float)data_1/data_2)*data_3;
```

콤파일러는 식 data_1/data_2 에서 류점인 data_1 변수를 계산하며 옹근수변수 data_2 를 류점수로 변환한다. 이 변환은 옹근수나누기가 진행될 때 정확성이 없어 지지 않는다는것을 보여 준다. 결과 류점수는 잘리우지 않으며 식의 나머지를 평가하는데 리용되게 된다.

우선권

복잡한 식을 고찰할 때 우선권의 개념을 리해하는것이 매우 중요하다. C언어는 우선권과 관련된 규칙을 가지고 있다. 이 규칙들은 식의 평가순서를 결정한다. 가장 높은 우선권을 가진 연산자를 제일 먼저 적용된다. 우선권이 같은 연산자는 놓인 차례로 적용된다.

표 22-4에는 높은 우선권순서로부터 우선권의 적용순서를 보여 주었다.

표 22-4 우선권순서에 따르는 C 연산자들

연 산 자	결 합 성
( )	왼쪽에서 오른쪽으로
한인수+, 한인수-, sizeof( ), ++, --, (type)	오른쪽에서 왼쪽으로
*, /, %	왼쪽에서 오른쪽으로
두 인수+, 두 인수-	왼쪽에서 오른쪽으로
=, + =, - =, * =, / =, % =	오른쪽에서 왼쪽으로

콤파일러는 가장 높은 우선권을 가진 연산자들을 먼저 처리한다. 콤파일러는 같은 우선권을 가진 두 연산자들이 처리되여야 할 때 우선권에 따라 어느것을 먼저 처리해야 하는가를 알고 있다. 련관성에 따라 콤파일러는 오른쪽의 연산자들로부터 시작할수도 있고 왼쪽으로 이동하여 처리할수도 있으며 또는 거꾸로 할수도 있다.

표의 제일 웃줄에 있는 연산자( )에 특별한 주의를 돌려야 한다. 이 연산자는 먼저 괄호안의 식에 대한 평가를 한다. 왜냐하면 이 식이 가장 높은 우선권을 가지고 있기때문이다.

```
total=2+8*8/2            /*결과가 34로 평가된다. */
```

웃 실례에서 연산자 *와 /는 같은 우선권을 가진다. 련관성은 이 식의 왼쪽에서 오른쪽으로 평가한다는것을 알수 있다.

연산자 *는 먼저 처리된다. 그리고 다음 연산자 /가 처리되며 64/2 는 32 로 된다. 2항연산자 +는 그 다음에 처리된다. 왜냐하면 +가 다른 연산자보다 낮은 우선권을 가지기때문이다. 따라서 total 은 34 로 된다.

괄호연산자가 리용되면 결과는 달라 진다.

```
total=(2+8)*8/2          /*결과는 40으로 된다. */
```

괄호는 가장 높은 우선권을 가지므로 그안에 있는 임의의 내용은 먼저 처리된다. 식 2+8 은 10 으로 된다. 다음 연산자 *와 /는 같은 우선권을 가지며 왼쪽에서 오른쪽으로 처리된다. 10*8 은 80 이고 80/2 는 40 이다. 이 식의 마지막결과는 앞의 실례에서와 다르다. 이것은 괄호연산자의 높은 우선권과 관계된다.

괄호들은 겹칠수 있다. 콤파일러는 항상 제일 안에 있는 괄호부터 처리하며 차례차례 처리하다가 가장 밖에 있는 괄호까지 이동한다. 이런 처리는 식의 마지막처리를 할 때까지 계속된다. 아래의 내용은 겹괄호를 가진 식이 어떻게 평가되는가를 론리적으로 보여 준다.

```
int a = 3, b = 2, c = 6, d = 2 ;
int e = 3, f = 4, g = 5 ;

total = ( a - ( b + ( ( c * d ) / e ) * f ) + g) - 1 ;
                         ↑
                    12로 평가됨.

total = ( a - ( b + ( 12 / e ) * f ) + g ) - 1 ;
                        ↑
                   4로 평가됨.

total = ( a - ( b + 4 * f ) + g ) - 1 ;
                    ↑
               18로 평가됨.

total = ( a - 18 + g ) - 1 ;
              ↑
         -10으로 평가됨.
```

```
total = - 10 - 1 ;
total = - 11 ;
```

순　환

지금까지는 C 언어프로그람작성에 필요한 기초적인 자료들을 보았으므로 더 복잡한 프로그람을 만드는데 필요한 조종명령들을 몇가지 고찰하기로 한다. C 언어는 조종명령문들을 가지고 있다.

이 부분에서 사용자는 몇개의 명령문들을 반복수행하는 방식에 대하여 고찰하게 된다. 이 방식이 바로 순환이다.

순환에는 2가지 형태 즉 while(while 순환은 **do while**이라고 부른다.)과 for가 있다. 순환과 순환에 대한 기본적인 조작에 대해서는 앞부분에서 이미 고찰되였다.

여기서는 C 순환문법에 대하여 고찰한다. 앞에서 서술한 순환을 리용하여 작업할 때 순환은 식의 귀환값에 관계된다. 이 값들은 True 혹은 False를 취한다. 여기서 True는 령이 아닌 수이며 False는 령인 수이다. 프로그람작성자는 프로그람에서 이 값들을 위한 지적자를 정의할수 있다.

For 순환

순환에서는 순환조종을 정확히 정의해야 순환을 얼마만큼 하였는가를 알수 있다. 이 명령문의 형식은 다음과 같다.

```
for (initialize expression; loop condition; update expression)
    statement;
```

세가지식 즉 초기화식, 순환조건식, 걸음식들은 순환을 위한 환경을 설정하고 조종한다. 그뒤의 명령들은 C명령문으로 이루어 진 순환의 본체이다.

순환의 본체가 한개 명령문이상으로 이루어 질 때 대괄호({ })를 리용하여 본체를 정의한다.

```
for (initialize expression; loop condition; update expression)
{
    statement;
    statement;
}
```

for 순환의 첫번째 성분을 **초기화식**(Initialize Expression)이라고 부른다. 이것은 순환을 시작하기전에 조종변수의 초기값을 설정하는데 리용된다. 초기화식의 실례로서 x=0;을 들수 있다. 변수 x는 순환이 시작되기전에 령의 값을 주게 된다. 초기화식은 순환의 첫 시작에서 한번 수행된다.

두번째 성분인 순환조건은 조건을 지정하거나 순환을 계속하는데 필요한 조건설정을

지정한다. 순환조건은 련관식들도 지정된다. 련관식은 비교연산자를 리용하여 변수들을 비교한다.

비교연산자들은 변수들, 상수들, 식들을 비교하게 하는 연산자이다. 표 22-5 에서 볼수 있는바와 같이 C 언어에는 6 개의 관계를 검열할수 있는 연산자가 있다.

표 22-5 비교연산자들

비교연산자	의미	실례
<	보다 작기	j<19
>	보다 크기	counter>10
==	같기	flag==DONE
!=	같지 않기	dog!=cat
<=	작거나 같기	temp<=freezing
>=	크거나 같기	air_miles>=premier

아래의 연산자들을 리용한 식들은 TRUE 인가 FALSE 인가를 평가한다. 비교연산자들은 산수연산자보다 더 낮은 우선권을 가진다. 여기서 주의할것은 대입연산자(=)와 비교연산자(= =)를 혼돈하지 말아야 한다는것이다.

```
count = = 2 ;
```

이 식은 계산값이 2 와 같은가, 같지 않은가를 검열하며 TRUE 인가, FALSE 인가를 검열한다.

```
count = 2 ;
```

이 식은 변수 count 에 값 2 를 준다. 비교연산자들중에서 순환을 위하여 어느것을 선택하겠는가 하는것은 계산실험으로 알수 있다.

순환조건은 초기화식다음에 검사된다. 순환조건이 FALSE 이면 순환은 순환본체의 명령문들을 수행한다. 순환조건이 TRUE 이면 순환은 정지되며 순환을 끝내고 순환본체다음에 있는 프로그람의 명령으로 이행한다.

세번째 성분인 증분식은 순환본체의 명령문들이 수행된 다음에 계산된다. 이 증분식은 조종변수들의 값을 변화시킨다. 증분식이 수행된 다음 순환조건은 순환이 수행되여야 하는가 수행되지 말아야 하는가를 다시 검사한다.

다음 프로그람토막은 순환에 대하여 설명한다.

```
int x;                        /*for 순환에서 리용된 조종변수는
                                x 이다. */

for (x=0; x<3; x++)           /*정확히 3 번 순환시키라*/
{
printf("%d", x);              /*조종변수 x 를 출력하라*/
}
```

```
printf("\n The loop is complete.\n");
```

웃 실례에서 x는 옹근수로 정의되며 for 순환에서 사용된다. 아래에서는 순환이 수행되는 방식을 설명하고 있다.

1. 순환이 시작되는 첫 순간에 초기화식은 x에 0을 준다.
2. 다음 순환조건은 x가 3보다 작은가를 검열한다. 이 결과는 TRUE이다.
3. printf( )함수는 화면에 0을 출력한다.
4. 증분식을 계산하고 x를 1만큼 증가시킨다. x값은 현재 1이다.
5. 순환조건은 x가 3보다 작은가를 검열한다. 이 결과는 TRUE이다.
6. printf()함수는 화면에 1을 출력한다.
7. 증분식을 계산하고 x를 1만큼 증가시킨다. x값은 현재 3이다.
8. 순환조건은 x가 3보다 작은가를 검열한다. 이 결과는 TRUE이다.
9. printf()함수는 화면에 2를 출력한다.
10. 증분식을 계산하고 x를 1만큼 증가시킨다. x값은 현재 3이다.
11. 순환조건은 x가 3보다 작은가를 검열한다. 이 결과는 FALSE이다. 모든 순환을 중지하고 끌낸다.
12. 순환본체의 바로 아래에 있는 printf()함수는 화면에 "The loop is complete"를 출력한다.

While 순환

while 순환은 사용자가 이미 결정한 어떤 조건이 일어날 때까지 while 순환본체의 수행을 반복하게 한다. 이 조건은 미정순환으로 알려 져 있다. while 순환은 이미 결정된 회수만큼 반복하지는 않지만 대신 이미 결정된 조건에 따라 순환을 계속한다.

while 순환문법은 다음과 같다.

```
while (loop condition)
        statement;
```

순환이 수행될 한개이상의 명령문을 가지고 있다면 순환의 본체를 대괄호({})안에 넣어야 한다.

```
while (loop condition)
    {
        statement_1;
        statement_2;
    }
```

이 명령을 수행할 때 괄호안에 있는 순환조건이나 식이 먼저 평가된다. 식의 결과가 TRUE 이면 while 순환의 본체가 수행된다. 순환의 본체가 수행된후에 식은 다시 평가된다. 식은 여전히 TRUE 이면 순환의 본체가 다시 수행된다. 식이 FALSE 이면 순환이 정지된다.

while 순환의 실례로서 다음의 원천코드토막은 수 5 까지 계산한다.

```
int  count=1;
while（count<=5）                    /*count 가 5 와 같거나 작을 동안 순환
                                       한다. */
{
    printf（"%d"，count);             /*count 변수값출력*/
    count++;                          /*count 변수값증가*/
}
```

결과는 다음과 같다.

```
1    2    3    4    5
```

while 순환은 식이 FALSE 로 평가될 때 정지된다. 여기서는 순환시간이 얼마나 되는가를 말해 주는것이 아니라 다만 순환이 "count 변수값이 5 보다 크면 정지하고 그렇지 않으면 순환하라"라는것을 말하여 주고 있다. 이것을 설명하기 위하여 아래에서 초기값이 3 과 같은 while 순환을 리용하자.

```
int  count=3;
while（count<=5）                    /*count 변수값이 5 와 같거나 작을 동안
                                       순환한다. */
{
    printf（"%d"，count);             /*count 변수값출력*/
    count++;                          /*count 변수값증가*/
}
```

결과는 다음과 같다.

```
3    4    5
```

이것은 초기값을 다른 수로 시작하였기때문에 다른 결과를 주는 while 순환이다.

Do While 순환

앞에서 론의된 두 순환들은 순환을 수행하기 위하여 전체 순환조건을 검사한다. 그러나 do while 순환은 먼저 순환을 수행하며 그다음에 순환조건을 검사한다. do while 순환의 문법은 다음과 같다.

```
do
{
    statement_1;
    statement_2;
}
while (loop condition);
```

이것의 실행은 다음순서로 한다. 순환의 본체가 먼저 수행되며 그다음에 순환조건을 검사한다. 순환조건이 TRUE 이면 순환의 본체가 먼저 수행되고 FALSE 이면 순환이 정지된다.

Break 명령문

순환의 본체안에서 어떤 조건이 생기면 순환을 그만두려고 하는 경우가 있다. break 명령문은 이러한 목적에 리용된다. break 명령이 수행되면 순환이 곧 정지되고 순환본체는 그대로 둔다. break 명령문은 다음과 같다.

```
break ;
```

Continue 명령문

continue 명령문은 순환을 결속 짓지 않는다는것을 제외하고는 break 명령문과 류사하다. 순환의 본체에서 continue 명령문은 그뒤에 오는 명령문을 계속 수행하도록 한다. 다른 한편 순환의 수행은 계속 된다. continue 명령문의 형식은 다음과 같다.

```
continue ;
```

론리연산자

론리연산자는 복합순환조건을 위한 합성련관검열을 위하여 사용한다. 표 22-6 에서 보여 주는바와 같이 C 에는 세가지 론리연산자가 있다.

표 22-6 론리연산자

연산자	의미
!	론리부정(NOT)
&&	론리적(AND)
||	론리합(OR)

론리연산자는 TRUE 혹은 FALSE 값을 준다. 론리부정은 연산인수가 TRUE 이면

FALSE 로 주고 FALSE 이면 TRUE 로 주는 연산인수가 하나인 연산자이다. 가령 옹근수 변수 z 가 령값(FALSE 값)을 가질 때 !z 는 TRUE 이다.

론리적과 론리합은 2 항연산자(두개의 연산인수를 리용)이다. 론리적은 두개의 연산인수가 둘 다 TRUE 이면 TRUE 를 주고 그렇지 않으면 FALSE 를 준다. 론리합은 어느 한 연산인수라도 TRUE 이면 TRUE 를 주고 그렇지 않으면 FALSE 를 준다.

실례로 다음의 합성관계식은 론리적을 리용하고 있다. 이 식은 두개 연산인수가 TRUE 이기때문에 TRUE 를 준다.

checking=100 이고 saving=100 이면 다음 명령문은 TRUE 이다.

```
checking>50  &&  savings<200
```

식의 두 쪽이 다 TRUE 이기때문에 론리적은 TRUE 로 평가된다. 두 연산인수중 하나라도 FALSE 이면 론리적연산자는 FALSE 로 된다.

론리합은 적어도 하나가 TRUE 이면 TRUE 로 된다. 실례로 다음 식은 TRUE 가 된다.

checking=100 이고 saving=100 이면 다음 명령문은 FALSE 이다.

```
checking>500  ||  savings<200
```

론리합을 리용한 웃 식의 두개 연산인수가 다 TRUE 이므로 TRUE 로 된다. 두개의 연산인수가 다 FALSE 이면 론리합은 FALSE 로 된다. 순환명령문에서 순환조건으로 론리연산자들이 있는 식을 리용할수 있다.

겹친 순환

순환들에서 하나의 순환안에 다른 순환이 겹놓일수 있다. while 순환이 for 순환안에 포함될수 있다. 외부순환이 한번 수행될 때 내부순환은 끝까지 수행된다. 조종은 내부순환을 다 수행한 다음 외부순환으로 되돌아 나가며 처리는 외부순환조건이 FALSE 가 될 때까지 계속된다. 아래에서는 겹친 순환에 대한 성질을 설명하는 C 원천코드토막을 제시하였다.

```
int outer, inner;
for  ( outer = 0; outer <= 4; outer++)
/*문자  'A'와 새행을 출력하는 외부순환*/
{
   printf( "%c\n", 'A');
   for ( inner = 0; inner < 4; inner++)
          /* 문자  'b'를 출력하는 내부순환*/
    {
     printf("%C", 'b');
     }
   printf ("\n");   /* b 행을 끝내고 새행 바꾸기*/
 }
```

이 코드토막의 출력은 다음과 같다.

```
A
b  b      b
A
b  b      b
A
b  b      b
A
b  b      b
```

외부순환이 한번 실행될 때 내부순환은 끝까지 모두 수행된다. 외부순환이 다시 한번 수행될 때 내부순환은 다시 시동되여 끝까지 수행된다. 겹친 순환들은 제일 바깥에 놓인 외부순환의 순환조건이 FALSE가 된 다음에 끝난다.

선　택

프로그람작성에서 또 하나의 중요한 성질은 조건에 따라 처리하는 기능이다. 이 부분에서 소개되는 조종명령들은 프로그람의 흐름을 조건에 따라 처리하도록 조종한다.

if 명령문

조건처리를 진행하는 명령문들중에서 먼저 if명령문에 대하여 보자.
if명령문의 형식은 다음과 같다.

```
if（expression）
    statement;
```

여러개 명령문을 수행하려면 if명령문의 본체에서 대괄호({})를 리용해야 한다.

```
if（expression）
{
    statement_1;
    statement_2;
}
```

if명령문에서는 "x의 값이 100이면 정확하다"라는것을 다음과 같이 써야 한다.

```
if (car==100)
    printf("x의 값이 100이면 정확하다. ￥n")
```

함수 printf( )는 식이 TRUE로 계산되면 한개 부분을 처리하고 식이 FALSE로 계산되면 또 다른 부분을 처리한다.

if else 명령문

if…else 명령문은 식이 TRUE로 평가되면 한 작용을 수행하고 식이 FALSE로 평가되면 다른 작용을 수행한다. 이 명령문의 형식은 다음과 같다.

```
if (expression)
    statement;
else
    statement;
```

여러개 명령문을 수행해야 한다면 대괄호({})를 사용해야 한다.

```
if (expression)
{
    statement_1;
    statement_2;
}
else
{
    statement_1;
    statement_2;
}
```

if…else 명령문의 실례는 다음과 같다.

x의 값이 100보다 크면 "100보다 크다. "는 통보를 출력하고 x의 값이 100보다 작으면 "100보다 작다. "는 통보를 출력하자.

```
if (x<=100)
    printf("100보다 크다. ￥n");
else
    printf("100보다 작다. ￥n");
```

겹친 if와 if else, else if

if 와 if…else 명령문들은 내부에 if 와 if…else 명령문을 포함할수 있다. 이때 명령문들의 의미가 정확하도록 조직해야 한다. if 와 그것에 대응하는 else 가 정확히 대응되지 않으면 혼돈될수 있다. 아래에 정확한 실례를 제시하였다.

```
if（expression_1）
{
    if（expression_2）
            statement_1;
}
else
{
    if（expression_3）
            statement_2;
}
```

statement_1 은 외부순환의 식 1 이 TRUE 로 계산되고 내부순환의 식 2 가 TRUE 로 계산될 때에만 수행된다.

statement_2 는 외부순환의 식 1 이 FALSE 로 계산되고 내부순환의 식 3 이 TRUE 로 계산될 때에만 수행된다.

else if 명령문은 겹친 명령문구조를 단순하게 할수 있다.

```
if（expression_1）
{
    if（expression_2）
            statement_1;}
elseif（expression_3）
            statement_2;
```

else 와 그것에 대응한 겹친 if 명령문은 한개 행에 써야 한다.

론리연산자

if 명령문은 두 값들중에서 어느 하나를 선택하는데 리용된다. 론리연산자는 이 장의 앞에서 설명한바와 같이 합성련관관계를 검열하는데 리용된다. 이에 대한 실례는 다음과 같다.

```
if（car <= 100 && car_year > 1992）
```

```
printf ("I?ll take it!\n");
```

이 실례는 car 가 100 보다 크지 않고 car_year 가 1992 보다 크면 식은 TRUE 로 된다.

Switch 명령문

Switch 명령문은 다중갈래를 조성하기 위하여 if…else 명령문을 변경한 형식이다. 이 명령문의 일반형식은 다음과 같다.

```
switch (expression)
{
    case value_1:
            statement;
            statement;
            …
            break;
    case value_2:
            statement;
            statement;
            …
            break;

    case value_n:
            statement;
            statement;
            …
            break;
    default:
            statement;
            statement;
            …
            break;
}
```

괄호안의 식은 value_1, value_2, …, value_n 과 비교된다. 이 값들은 상수이거나 상수식이여야 한다. 이 값들중에서 식과 같은것을 찾으면 그 갈래에 해당한 명령문들이 수행되고 같은것을 찾지 못하면 default 부분에 대응한 명령문들이 수행된다.

다음의 Switch 명령문을 리용한 실례는 승용차제작년도에 따라 승용차에 대한 통보를 출력한것이다.

```
switch    (car_year)
{
case      1999:
          printf("I' ll offer you $8, 000 for the car.\n");
          break;
case      1998:
          printf(I' ll offer you $6, 000 for the car.\n");
          break;
case      1997:
          printf(I' ll offer you $3, 500 for the car.\n");
          break;
case      1996:
          printf("I' ll offer you $2, 100 for the car.\n");
          break;
case      1995:
          printf("I' ll offer you $1, 000 for the car.\n");
          break;
case      1994:
          printf("I' ll offer you $800 for the car.\n");
          break;
case      1993:
          printf("I' ll offer you $400 for the car.\n");
          break;
default:
          printf("No thank you, the car is too old.\n");
          break;
}
```

이 실례에서는 조건을 비교하여 같게 되는 명령문을 수행한다. 승용차생산년도가 1993년 이전이면 비교하지 않으며 switch명령문은 default부분을 수행한다.

함 수

함수들은 C 언어프로그람에서 기본블로크이다. 이미 화면에 인쇄하기 위한 함수 printf( ) 혹은 건반으로부터 자료를 읽기 위한 함수 scanf( )와 같은 일부 함수들을 보았다. 그리고 모든 C프로그람이 main( )이라는 함수로 시작된다는것을 알고 있다. 함수는 이미 만들어 놓고 사용한다.

다음의 경우에 함수들을 만들수 있다. 첫째로, 연산이 프로그람안에서 여러번 리용되

는 경우에 함수로 만들수 있다. 둘째로, 연산이 두개이상의 프로그람들사이에서 리용되는 경우에 함수를 만들수 있다. 셋째로, 연산이 하나의 결과를 주는 경우에 함수를 만들수 있다. 넷째로, 한번만 리용된다고 해도 분리해야 할 과제인 경우에 함수로 만들수 있다. 다섯째로, 프로그람이 너무 긴 경우에 보기 편리하게 하기 위하여 함수로 만들수 있다.

프로그람에서 함수들을 리용하려고 하면 함수들을 먼저 정의하고 그다음에 함수를 호출한다. 함수는 개별적인 원천코드들로 정의되여야 하며 함수를 리용할 때에는 그 함수의 이름으로 호출하여야 한다.

많이 사용하는 함수들은 C 콤파일러의 표준서고에 들어 있다. 표준함수를 리용하면 프로그람작성을 더 쉽게 할수 있다.

함수호출

함수호출형식은 다음과 같다.

```
function_name（arg_1，arg_2，…）
```

어떤 연산이 필요할 때 함수호출은 그 연산을 위하여 작성한 함수이름으로 하여야 한다.

함수를 호출하려면 함수이름을 필요한 위치에 지적하여야 한다. 함수이름은 변수이름과 같은 규칙으로 사용된다. 함수이름은 문자로 시작하여 문자, 수자 또는 지정된 기호들로 구성되여야 한다. 인수들은 함수뒤의 괄호안에 기입한다. 인수들은 값과 관련되며 함수에서 리용된다. 인수들은 상수, 변수, 식들이 될수 있으며 반점으로 분리된다. 함수에서 값들은 인수를 통해 받지 않으려면 변수들을 생략할수 있다. 함수는 괄호안에 있는 인수들을 포함하여 함수이름으로서 다른 함수와 련결된다.

함수를 호출할 때 다음의 두가지 처리를 한다. 첫째로, 함수가 실행된다. 둘째로, 값을 호출한 함수에 귀환한다.

함수정의

함수정의형식은 다음과 같다.

```
return_type function_name（ parameter list）
{
    statement(s);
}
```

함수는 다음의 네가지 부분을 포함한다.

1. **귀환형**(Return Type) 함수가 값을 되돌려 줄수 있기때문에 그 값의 자료형을 정의해야 한다. 함수에 의하여 아무런 값도 귀환하지 않으려면 void 자료형을 리

용하여야 한다. 귀환형의 실례는 다음과 같다.

```
int price_service (int service_type, float cost_per_service)
{
        /*계산과정명령문들*/
}
```

2. **함수이름**(Function Name) C 언어에서 함수이름의 최대길이는 31 개 문자이다. 함수이름은 더 길수 있지만 대부분 콤파일러들은 첫 31 개 문자만 인식한다.

```
int price_service (int service_type, float cost_per_service)
{
  /*계산과정명령문들*/
}
```

3. **파라메터목록**(Parameter List) 파라메터는 선택적이지만 괄호는 있어야 한다. 파라메터와 인수들은 같은 위치에서 서로 대응된다. 인수는 함수정의에서 파라메터에 대응하는 함수호출명령에서의 값이다. 파라메터들은 함수를 정의할 때 괄호안에서 반점으로 분리된 변수들이다. 한개 파라메터들은 자료형과 변수이름으로 이루어 진다. 매개 함수호출명령들에서 인수는 개수, 자료형 그리고 함수정의에서 파라메터들의 순서와 일치되여야 한다.

```
int price_service (int service_type, float cost_per_service)
{
  /*계산과정명령문들*/
}
```

4. **함수본체**(Function Body) 함수본체는 대괄호({})안에 포함된다. 본체안에는 필요한 연산을 수행할 명령문들을 써넣어야 한다.

```
int price_service (int service_type, float cost_per_service)
 {
  /*계산과정명령문들*/
 }
```

함수를 호출하면 그 이름으로 정의된 함수가 대응된다.

함수정의에서 사용자는 함수(함수이름)호출방법과 함수를 수행하고 함수에서 귀환할 값, 어떤 자료형이 필요한가 하는것을 규정하게 된다.

함수정의본체는 이런 함수의 원천코드들을 포함한다. 파라메터목록에서의 파라메터들은 인수들이 어떤 자료형을 가져야 하는가 또한 어떤 순서로 그 개수가 얼마인가를 지적해 준다. 함수를 호출하면 함수정의파라메터에 대응한 인수들에 값을 넘겨 주고 함수정의에서 본체부분을 찾아 실행시킨다. 그림 22-5 에서는 함수의 구조를 보여 준다.

```
main()
{
        int flag;        /*함수호출이 성과적으로 진행되면 령을 귀환하고
                           오류가 있으면 령이 아닌 값을 귀환 */

        float service_cost;
        ...
        flag = price_service(1, service_cost);  <---- price_service()의 함수호출과 인수들
        ...
        return;
 }
 int price_service(int service_type, float cost)  <---- price_service()의 함수호출과 파라메터들
 {
        /*price service()의 명령문들 */
        ...
        return (0)
 }
```

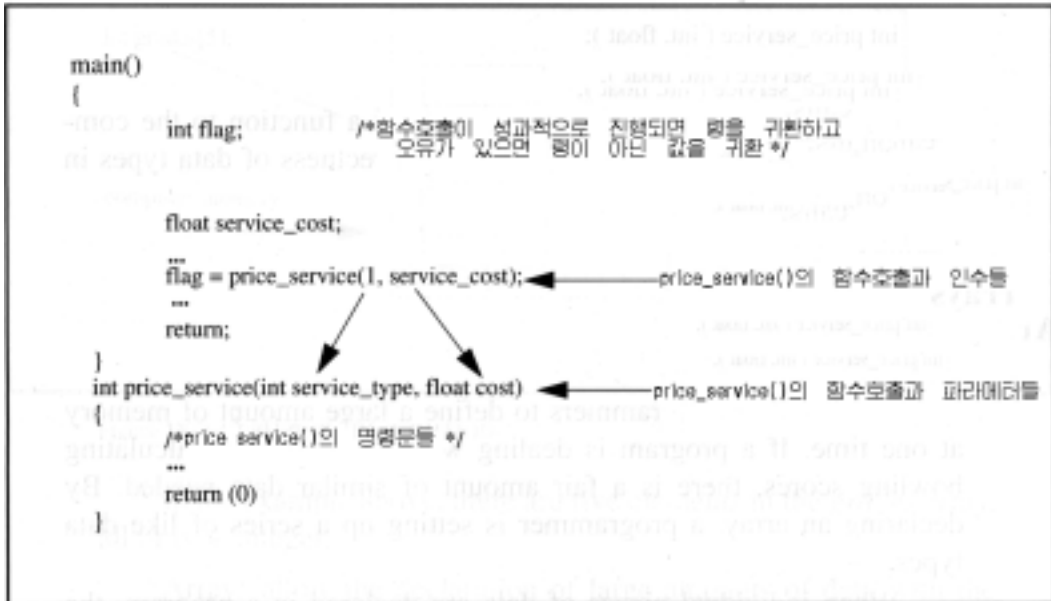



그림 22-5. 함수의 구조

원형

함수를 호출할 때 콤파일러는 인수들과 함수정의에 있는 파라메터들과 련관을 보장한다. 콤파일러는 첫번째 인수를 첫번째 파라메터에 주며 두번째 인수는 두번째 파라메터에 준다. 인수들의 개수는 파라메터의 개수와 일치되여야 한다. 이 개수들이 같지 않으면 오유가 발생된다.

원형들은 콤파일러가 인수들의 형과 파라메터형이 일치되지 않는것을 검사하는데 리용된다. 함수원형은 반두점(;)으로 끝나는 함수정의에서의 첫행이다. 웃 실례에서 함수 price_service( )의 원형은 다음과 같다.

```
int price_service (int service_type, float cost);
```

변수이름들은 원형에서 필요하지 않으면 지적하지 않을수도 있다.

```
int price_service (int, float);
```

원형은 콤파일러에 함수에 대한 자료를 줌으로써 함수호출에서 자료형들의 정확성을 검열하게 한다.

배 렬

배렬은 프로그람작성자가 한번에 많은 용량의 기억을 정의할수 있게 한다. 프로그람을 통하여 경기점수를 계산하거나 달력과 같은 날자를 계산할 때에는 많은 자료들이 필요하다. 프로그람작성자는 배렬을 소개하여 자료형이 같은 렬들을 정의할수 있다.

프로그람에서 자료를 소개할 때 그림 22-6에서 보여 주는것처럼 자료는 고정된 기억기에 보관되는것이 아니라 임의의 곳에 보관될수 있다.

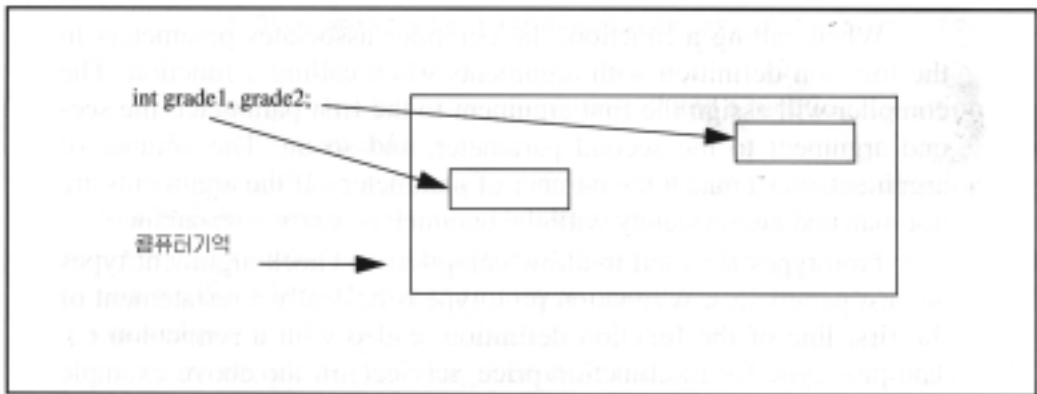



그림 22-6. 임의의 곳에 정보보관

배렬을 사용하면 그림 22-7에서 보여 준바와 같이 자료들이 련이어 보관된다.

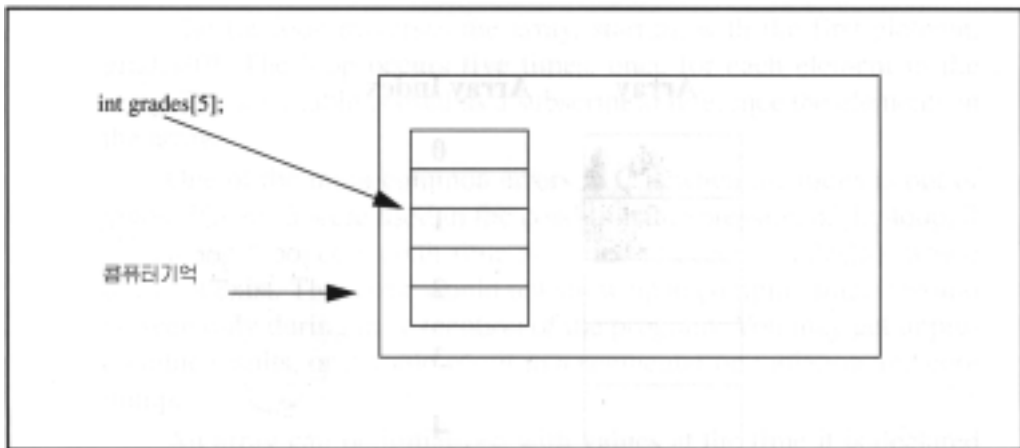



그림 22-7. 자료들의 배렬

웃 실례에서 grades배렬에 5개의 원소들이 있는데 이것들은 모두 옹근수형이다.

배렬과 같은 자료형을 리용하면 많은 량의 자료정의를 한개 명령문으로 할수 있다. 이 배렬은 매개 자료형과 변수들에 대하여 개별적인 명령으로 처리하는것보다 훨씬 쉽다. 배렬을 리용

하면 자료처리가 단순하고 혼란이 적어 진다. 배렬을 리용하면 또한 원천코드를 보는것도 쉽다.
배렬소개형식은 다음과 같다.

```
data_type  array_name  [array_size];
```

배렬의 크기를 지적하는 수는 옹근수만이 될수 있다. 상수들, 기호상수들 그리고 상수식들을 지적할수 있으나 이 상수식들의 결과는 옹근수값이여야 한다.

배렬 배렬첨수

0
1
2
3
4

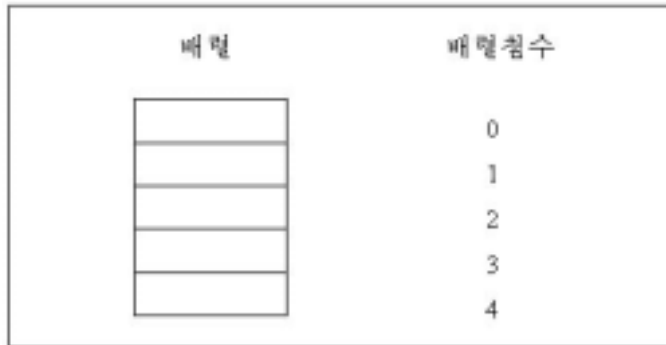



그림 22-8. 배렬

배렬의 매개 성분을 **원소**(Element)라고 부른다. 매 원소는 배렬에서 같은 자료형으로 된다.

배렬에서 원소들은 옹근수첨수들에 의하여 서로 련관을 가진다. 이 배렬원소들에서 첨수는 령으로부터 시작된다. 그림 22-8에서는 첨수있는 배렬에 대하여 보여 준다.

배렬에서는 원소를 지적하기 위하여 첨수를 사용한다. 첨수는 원소의 이름뒤의 중괄호([ ])안에 넣는다. 통보에 접근할 때 첨수는 배렬의 위치를 지적한다. 아래에 있는 코드토막은 grades라는 배렬을 정의하고 있으며 배렬에서 첫번째 및 세번째 원소에 값들을 넣는다는것을 보여 주고 있다.

```
int  grades  [5];

    …
    grades  [0]=100;
    grades  [2]=87;
```

순환은 배렬을 정의할 때 아주 편리하다. 프로그람에서 5명의 학생들에 대한 점수를 grades라는 배렬로 정의하였으면 배렬에서 grades의 합은 for순환으로 다음과 같이 쓸수 있다.

```
for (x=0;  x<5;  x++)                /*5번  순환*/
sum+=grades[x];            /*sum=  sum+grades[x]를  간단하게  쓴것*/
```

for 순환은 배렬에서 첫 원소인 grades[0]부터 시작한다. 순환은 배렬에서 매 원소를 차례로 더하여 5번 진행한다. 변수 x는 배렬에서 원소들을 련결하기 위한 보조변수처럼

사용된다.

C 언어의 일반적으로 나타나는 오유중에서 하나는 첨수가 한계밖에 나가는것이다. 순환의 조건식에서 x<=5 가 사용되였다면 6 번 반복되면서 grades[5]에 접근하려고 한다. 이 오유는 콤파일러될 때에는 나타나지 않으며 프로그람을 수행할 때에만 볼수 있다. 이 경우에 사용자는 예견할수 없는 결과를 얻으며 오유로 하여 토막위반과 같은 혼란이 생길수 있다.

아래의 배렬에서 초기값들을 정의하는 실례를 준다.

```
int grades[5]={100, 98, 97, 87, 99};
```

콤파일러는 값을 배렬한다. 사용자는 배렬에 있는 원소개수보다 적은 값을 초기화할수 있다.

다차원배렬

지금까지 고찰한 배렬은 1 차원과 련관된 행배렬이였다. 배렬은 다차원일수도 있다. 다차원배렬에서 가장 일반적으로 리용하는것중의 하나인 그림 22-9 는 행렬이나 표와 같은 2 차원배렬에 대하여 보여 준다.

3 × 5 행렬을 다음과 같이 정의할수 있다.

```
int array_m[3] [5];
```

사용자가 행이나 렬이라는 술어로 표현한 첫째 중괄호안의 수는 렬의 개수를 나타내며 둘째 중괄호안의 개수는 행의 수를 나타낸다.

원소들을 지적하기 위하여 두개의 첨수들을 리용한다.

그림 22-9. 다차원배렬

array_m[0][4]는 첫행에서 마지막렬의 원소인 오른쪽 웃구석에 있는 원소 9 을 지적한다.

함수에 배렬을 전달

인수로서는 원소의 값이나 배렬전체를 보낼수 있다.
한개 원소를 보내기 위해서는 다음 실례에서처럼 함수호출과 정의를 할수 있다.

```
#include <stdio.h>
void print_grade(int);        /* 함수원형*/

main( )
{
    int grades[5] = (100, 8, 97, 87, 99);          /*배렬소개*/
    …
    print_grade( grades[0] );
    /* 배렬 grades 의 첫 원소를 함수의 인수에 보내기*/
    …
    return;
}
…
void print_grade( int a_grade )
{
    printf("The grade is %d.\n", a_grade);
}
```

프로그람의 출력은 다음과 같다.

```
The grade is 100.
```

함수에로 전체 배렬을 보내려면 중괄호 ([ ]) 혹은 다른 표시를 하지 않고 배렬이름만 쓰면 된다. 아래에서는 함수 minimum_grade( )을 통하여 전체 배렬을 함수에 보내는 것을 보여 주었다.

```
lowest=minmum_grade(grades);
```

함수정의는 다음과 같이 한다.

```
int minimum_grade(int all_grades[5])
{
    /*명령문들*/
    …
    return (minimum);
}
```

함수정의에서는 배렬의 크기를 반드시 지적하지 않아도 된다. 함수정의에서 배렬은 int all_grades[ ]로 소개할수 있다.

배렬은 유용한 구조이다. 이 부분은 초학자들에 도움을 주기 위하여 서술하였다.

기 호 렬

이 장의 앞부분에서 이미 고찰한것처럼 기호렬들은 문자들로 단어나 문장들을 만들게 한다. 기호렬은 자료형들에서 기본형으로 되지는 않지만 문자렬의 특수형태로 된다.

기호렬과 문자렬이 같은것은 아니다. 문자렬은 Null 값('￥ 0')으로 끝나는 기호값들의 렬을 포함하는 문자배렬이다. 문자배렬은 문자렬을 포함하지 말아야 한다. 왜냐하면 문자렬이 문자배렬의 값일수 있기때문이다.

다음의것은 문자배렬이다.

```
char array_of_chars [5]={' h ', ' e ', ' l ', ' l ', ' o '};
```

다음의것은 배렬에 문자렬을 포함한다.

```
char string_array [6]="hello";
```

기호렬은 기호렬상수 혹은 기호렬배렬으로 이루어 질수 있다. 기호렬상수는 문자들의 렬을 웃두빗점(" ")안에 포함한다. 기호렬상수는 끝에 빈 문자를 포함한다.

기호렬배렬은 기호렬을 포함하는 문자배렬이다. 4개 문자를 포함하는 문자배렬은 char letters[3]으로 소개할수 있다. 4개문자를 포함하는 기호렬은 char word[4]로 소개할수 있다. 문자배렬과 기호렬은 둘 다 4개의 문자들을 포함하지만 기호렬은 **Null 값**(Null Value)을 더 포함한다.

기호렬은 다음과 같이 한개 명령문에서 기호렬상수를 소개하고 초기값을 줄수 있다.

```
char student_name [10]="Carly";
```

기호렬은 문자배렬을 그대로 리용하지 않을수도 있지만 기호렬안에는 빈 기호가 있어야 하고 매 기호들은 련결되여야 한다.

문자배렬을 소개할 때에는 같기를 표시하는 대입연산자를 리용하여서만 문자렬에 문자렬상수를 초기값으로 줄수 있다.

```
char student_name ="Carly";
```

아래의 방법에 의해서는 프로그람본체에서 문자렬이나 기호렬에 초기값을 주고 값을 변경시킬수 없다.

다음의것은 허용되지 않는다.

```
student_name = “Carly”;
```

이것을 실행하기 위하여 C에는 strcpy( )라고 하는 서고함수가 있다. 기호렬표준함수

들은 콤파일러에 있다. 기호렬을 옳게 복사하려면 아래의 명령문을 사용한다.
아래의것은 허용되는 실례이다.

```
strcpy(student_name, "Carly");
```

이 함수의 원형은 머리부파일 string.h 에 있다. 임의의 기호렬서고함수를 리용할 때에는 반드시 다음의 처리지적자를 원천파일의 꼭대기에 포함시켜야 한다.

```
#include <string.h>
```

이밖에도 이 절에서 취급하지 않는 더 많은 기호렬함수들이 있다는것을 지적해 둔다.

구 조

묶음은 같은 자료형의 정보를 같은 그룹으로 만들며 이름을 리용하여 내용들이 련관을 가지게 한다.

C 도 구조라고 부르는 자료들을 묶을수 있는 도구를 제공한다.

구조들은 서로 다른 자료형들로 자료부류를 만들수 있는 방법을 주기때문에 단순하게 조종할수 있다. 무엇때문에 구조를 리용하는가 하는것은 여러가지로 설명할수 있다. 구조는 프로그람작성자가 공통적인 기억구조안에서 공통이름을 가진 련관된 자료항목들을 론리적으로 조직하여 준다. 실례로 종업원들을 관리하는 개별자료들로 그룹을 형성하는것을 둘수 있다. 매 종업원은 이름과 주소, 생활비와 수입, 직업 등을 가지고 있다. 매 자료부분은 한 종업원에게 론리적으로 련관되여 있지만 그것은 서로 다른 자료형으로 이루어 져 있다.

구조는 또한 함수안에 들어 가는 인수의 개수를 최소화하게 해준다. 함수가 종업원의 자료를 출력하려고 한다면 매개 자료형은 이름, 주소, 생활비 등과 같은 인수를 요구할것이다. 이 모든 자료가 한개 구조로 그룹을 형성하면 그 구조만이 함수에 들어 간다. 구조는 고찰하기 편리한 좋은 틀을 가지고 있다. 프로그람에서 모든 종업원자료를 하나의 자료구조로 묶으면 찾기 쉽다.

구조는 다음과 같은 형식에 의하여 정의된다.

```
struct tag
    {
    member_list;
    } name_list;
```

실마리어 struct 는 콤파일러에 이것이 대괄호({})안에 있는 자체성원목록을 포함하는 구조의 정의라는것을 알려 준다.

member_list 는 구조의 부분인 다른 자료형들의 자료를 포함하며 다른구조들도 포함한다.

tag 는 구조를 가리키는 식별자이거나 이름이다. 구조의 이름은 구조의 전개나 설계에 대하여 설명한다. tag 는 그 구조의 이름을 지정하는데 리용된다.

name_list 는 구조변수를 위한 식별자이거나 이름들의 목록이다. 다시 말하여 이 구조변수이름들은 name_list 에 대하여 제시한다. 구조의 변수이름은 구조이름의 설계에 따라 배치된 콤퓨터의 기억을 나타낸다.

구조는 두 단계로 정의된다. 즉 첫째로, 구조의 전개나 구조이름이 정의된다. 둘째로, 변수가 정의되고 기억이 할당된다. 이 단계들은 두 단계에서 또는 같은 명령문에서 실현될수 있다. 다음실례는 두 단계에서의 구조에 대한 정의를 보여 준다.

```
struct  employee_jobs
{
    char  job_title  [30];
    float  monthly_salary;
    int  job_level;
};
```

이것은 구조안에 넣을수 있는 모든 원소들을 포함한다. 이 정의는 변수이름을 포함하지 않는다. 이 정의는 변수이름을 포함하지 않는다. 이것은 다음 형식에서 employee_job 라는 이름에 의하여 수행된다.

```
struct  employee_jobs  manager;
```

변수관리자는 struct employee_jobs 라는 형이다.

다른 employee_jobs 형들은 다음형식에 의하여 정의할수 있다.

```
struct  employee_jobs  analyst;
struct  employee_jobs  ceo;
```

구조안에 있는 원소를 표현하는 방법은 조금씩 다르다. 점을 찍은 다음에 구조변수이름을 쓰는 방법을 리용하고 그 다음에 성원이름을 쓴다. 다음 명령문으로 ceo 의 생활비를 설정한다.

```
ceo.monthly_Salary=60000.00;
```

구조는 그전체가 한개 인수로 함수에 들어 갈수 있다. 그림 22-10 으로 제시된 실례를 보자.

함수 time_in_minutes 에 넘겨 진 time_now 구조가 함수정의에서는 변수 t 로 정의되였으며 함수본체안에서는 t 로 쓰인다. 또한 자료는 time 구조에서만 읽을수 있다. time 구조의 내용은 변하지 않는다. time_in_minutes 함수가 time 구조의 내용을 변경시키지 않으면 main( )함수에서 나타나지 않는다. 왜냐하면 구조의 사본이 함수에 들어가기때문이다. time_in_minutes( )함수에서 구조가 변화되면 main( )함수에 귀환될 때 사본이 제거된다.

```
struct time
{
        int hour;        <------------------  구조본보기소개
        int minute;
};

int time_in_minutes ( struct time t );   <------  함수설계

main()
{
        struct time_now;    <--------------  구조변수소개
        int minutes;

        time.hour = 10;
                             <--------------  구조변수에 값주기
        time.minute = 15;

        minutes = time_in_minute ( time_now );   <---  변할수 있는 구조로 함수를 호출

        printf("%d hours and %d minutes comes to a total of %d minutes\n",
                time_now.hour, time_now.minutes, minutes);

    return;
}

/*분으로 총 시간을 계산하는 함수*/

int time_in_minutes ( structure time t )   <------  함수소개는 구조를 파라메터로 가진다.
 {
    int total_minutes;


    total_minutes = t.hour*60;   /*분을 첨가*/

    return ( total_minutes );

 }
```

그림 22-10. 구조를 함수에 전달

작용구조에 넘어 가 함수범위내에서 내용을 변경시키기 위해서는 지적자를 리용하여야 한다. 지적자는 다음부분에서 론의하게 된다. 구조는 구체적으로 론의할수 있다.

이 부분에서는 구조에 대하여 간단히 론의하였다. 구조에 대한 그이상의 상세한 실례와 자료를 보려면 이 장의 마지막에서 지적하는 도서들을 보기 바란다.

지 적 자

마지막으로 지적자에 대하여 고찰한다. 지적자들은 C 에서 가장 중요한것들중의 하나로서 기교 있게 리용할수 있다. 지적자들은 배렬원소들을 처리하거나 함수범위내에서 인수들이 변경되도록 하는데서 인수들대신에 리용할수 있다. 이 지적자들은 변수크기형식으로 소개되지 않는다. 이 지적자들은 함수본체안에서 처리된다.

지적자는 어떤 값의 주소를 포함하는 장소이며 기억기에서 또 다른 곳을 지적하기도 한다. 지적자의 내용은 자료가 다른 자료를 처리하는데 리용될수 있다는것만을 제외하면 이미 고찰한 옹근수자료와 류사하다.

지적자들은 콤퓨터기억안에서 상수나 변수의 형태를 취할수 있다. 지적자변수들은 정의되거나 초기화될수 있다. 지적자변수값들은 변화시킬수 있다. 지적자상수값은 우리가 고찰한 다른 상수들처럼 변경될수 없다.

지적자변수는 다음의 형식으로 정의할수 있다.

```
data_type * pointer_name;
```

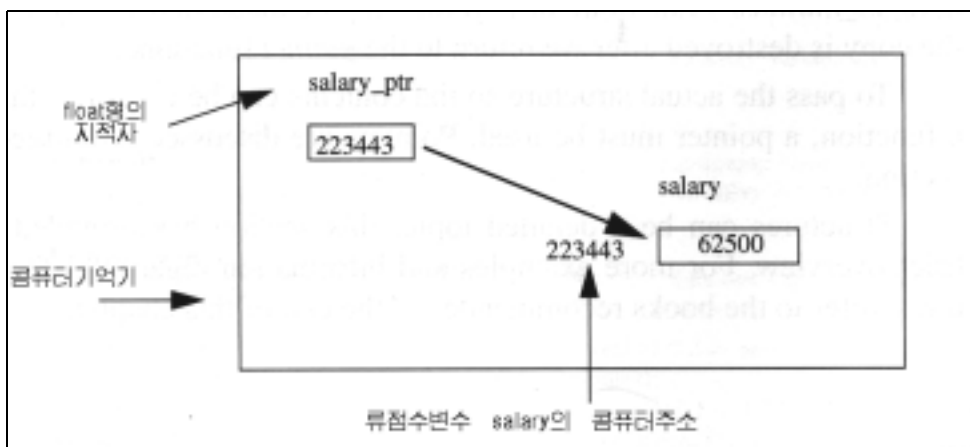



그림 22-11. 지적자

data_type 는 앞에서 본 자료형과 같다. 자료형은 어떤 자료형이 어느 지적자에 의하여 표시되겠는가를 지적한다. 지적자를 float 로 지적하려면 지적자를 float * salary_ptr;와 같이 소개하여야 한다. 그림 22-11 은 이 개념에 대하여 설명해 주고 있다.

지적자에 의하여 정의된 변수는 " 지적하는 곳"의 변수주소를 포함한다. 지적자 salary_ptr 는 그 지적자가 float 형의 변수로 지적되였으므로 float 형으로 소개되여야 한다.

지적자연산자

지적자와 함께 자주 쓰이는 두개의 연산자가 있다. 주소연산자(&)는 주소를 얻을수 있게 한다. 지적자연산자(*)는 주소위치에서 값을 처리할수 있게 한다. 이 연산자들은 서로 대응되여 수행된다. 해당한 위치를 지적하면 그것을 처리한다. 지적자에 값을 줄 때에는 주소연산자(&)를 리용한다. 지적자를 리용하는 자료를 처리할 때에는 지적연산자(*)를 리용한다.

그림 22-12 는 지적자의 리용을 설명한다.

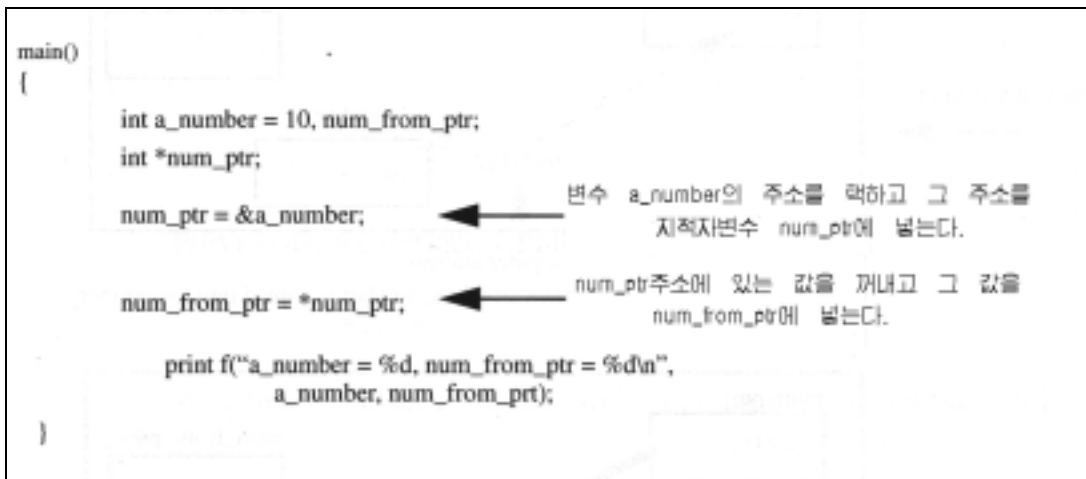



그림 22-12. 프로그람에서 지적자연산

그림 22-13 은 프로그람의 도형적설명으로 된다.

printf( )명령문은 다음과 같은 결과를 준다.

a_number = 10, num_from_ptr = 10

여기서 num_ptr 에 있는 값을 출력하면 결과는 a_number 의 변수주소로 된다.

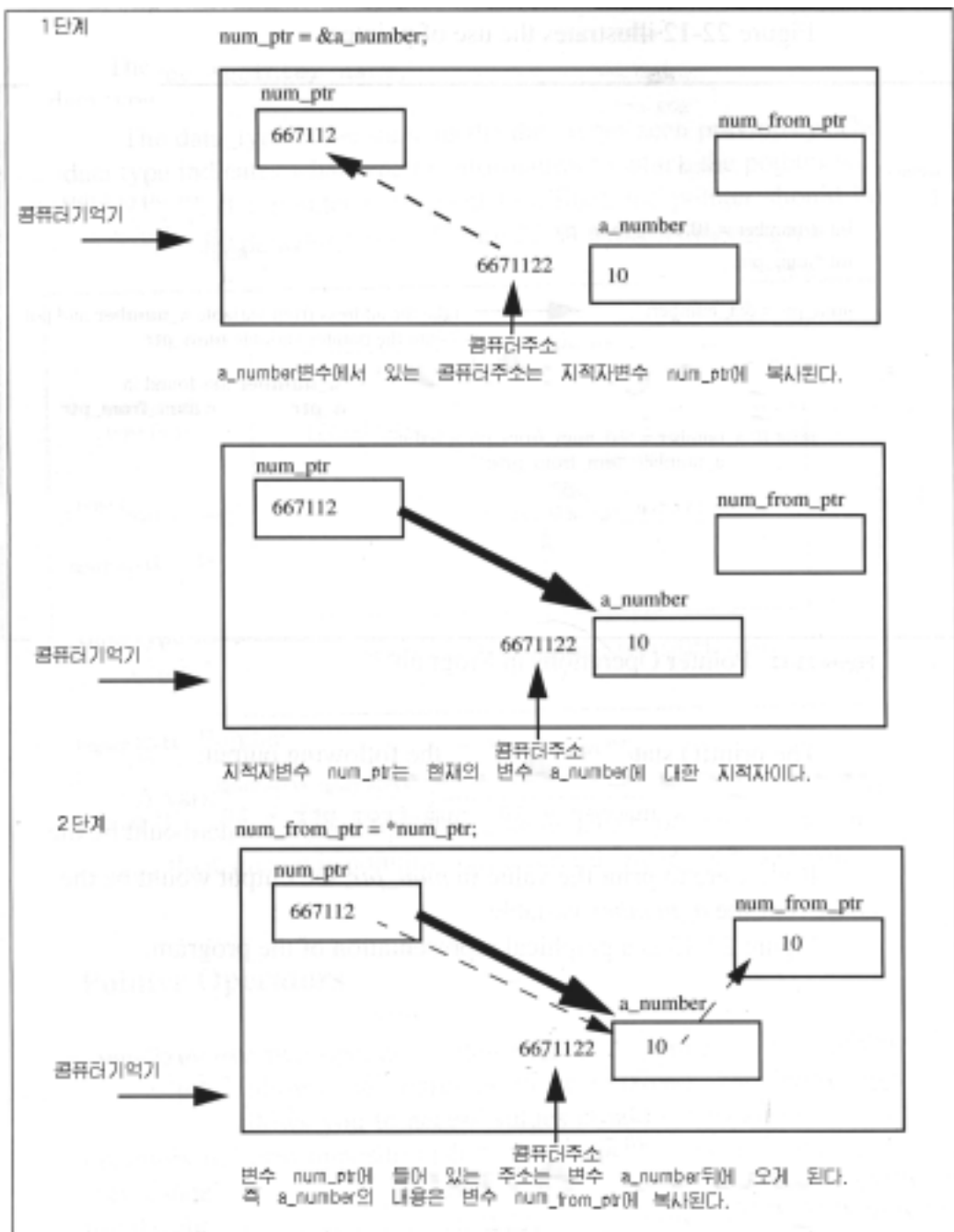



그림 22-13. 프로그람의 지적자연산의 도식표시

지적자와 구조

지적자들을 구조로 지적할수 있다. 다음의 구조에 대한 소개를 고찰하자.

```
struct time
{
    int hour;
    int minute;
};
```

앞에서는 struct time 형의 변수를 정의하였다.

```
struct time time_now;
```

우리는 또한 변수를 struct time 변수의 지적자로 정의하였다.

```
struct time *time_pointer;
```

지적자변수 time_pointer 의 구조변수 time_now 의 주소를 다음의 명령을 리용하여 줄수 있다.

```
time_pointer=&time_now;
```

값주기를 수행한 다음에는 time_pointer 의 time 구조의 성원들을 처리할수 있다.

```
(*time_pointer).hour=16;
```

구조는 또한 다음의 실례에서와 같이 지적자를 포함할수도 있다.

```
struct time
{
    int hour;
    int minute;
    int *seconds_ptr;
}
```

time 구조는 second_ptr 지적자를 가질수 있으며 초로 지적되는 수자를 포함하는 옹근수로 지적된다.

지적자와 함수

지적자들도 함수에 인수로 전달될수 있다. 이런 경우에 이미 고찰한바와 같이 그 변수의 사본이 전달되는것이 아니라 실제변수의 지적자가 전달된다. 만일 그 변수의 내용이 함수내에서 변경될 때 그 내용들은 그대로 남아 있는다.

함수호출에 있는 인수들과 함수정의와 원형에 있는 파라메터들은 지적자를 받아 들이도록 정의되여야 한다.

두개의 옹근수지적자를 받아 들이도록 정의된 한개 원형은 다음과 같다.

```
int time_in_minutes（ int *hour_ptr，int *minute_ptr）;
```

함수정의는 다음과 같이 진행한다.

```
int time_in_minutes（ int *hour_ptr，int *minute_ptr）
{
   /*명령문들*/
}
```

아래의 코드토막은 옹근수지적자를 요구하는 함수의 호출을 설명한다. 이 실례에서 변수들은 옹근수로 정의되여 있다. 함수를 리용하기 위해서는 변수주소들의 주소연산자(&)를 리용하여 함수에로 전달하여야 한다.

```
int hour，minute，total_min;
hour=10;
minute=33;
total_min=time_in_minutes（&hour，&minute）;
```

아래의 실례에서는 추가적인 변수들이 옹근수지적자로 정의되여 있다. 옹근수변수들은 값을 받고 그 변수의 주소는 옹근수지적자변수에 의하여 지정된다. 옹근수지적자변수가 이미 주소를 포함하고 있기때문에 그 주소는 임의의 지적자연산이 없이도 리용할수 있다.

```
int hour，minute，total_min;
int *hour_ptr，*minute_ptr;
hour=10;
minute=33;
hour_ptr=&hour;
minute_ptr=&minute;
total_min=time_in_minutes（hour_ptr，minute_ptr）;
```

지적자변수는 구조지적자에 의하여 정의될수 있다. 구조의 주소는 주소연산자(&)를 리용하여 함수에 보낼수 있거나 구조지적자변수를 리용하여 보낼수 있다.

구조지적자의 성원변수들을 련결하는것은 구조에서 직접 성원값들을 련결하는것과는 다르다.

구조에서 직접 성원변수들을 련결하는것은 구조변수이름과 성원변수이름사이에 (.)을 주는것으로 한다. 우리는 앞에서 아래의 코드토막과 류사한 실례를 보았다.

```
struct  time
{
    int  hour;
    int  minute;
};
main（ ）
{
    …
    struct  time  mytime;
    …
    mytime.hour=9;
    mytime.minute=45;
    …
}
```

구조지적자에서 성원함수들을 련결할 때에는 값주기기호대신에 구조지적자의 변수이름과 성원변수이름사이에 지적자기호(− >)를 사용한다. 이 기호는 두문자의 결합 즉 부수기호(−)와 크기부호(>)의 결합이다.

그림 22−14 는 구조와 지적자들에서 취하는 성원변수들을 설명하는 실례이다. 실례에서 time_in_minutes( )함수는 구조 time 지적자를 접수하도록 변경되였다.

main 함수에서는 시간구조지적자 time_ptr 가 정의된다. time_ptr 의 성원변수들을 설명할 때에는 점(.)이 사용되지 않지만 - >부호가 리용된다.

구조지적자를 접수하도록 time_to_minutes( )함수가 변경되였기때문에 성원변수들을 처리하기 위해서는 함수안에서 지적자기호(- >)를 리용해야 한다. 또한 함수호출은 구조변수를 리용할 때 변화된다. 현재 변수가 구조지적자를 접수하기때문에 구조변수는 주소지적자를 리용하여 그 주소를 보내야 한다. 구조지적자를 보낼 때에는 지적자연산자를 요구하지 않는다. 어떤 방식으로 정의된 함수안에서 성원변수들의 내용이 변화된것은 함수에서 벗어 난 다음에도 보존된다.

```
struct time
{
        int hour;
        int minute;
};

int time_in_minutes ( struct time * );          /*  함수원형            */

main()
{

    struct time mytime;                     /* 변수를  struct time형으로  소개     */
    struct time *time_ptr;                  /* 지적자를  struct time형으로  소개   */
    int total_min;


   mytime. hour = 9;                        /* 성원변수에  자료대입              */
   mytime.minute = 45;



   time_ptr = &mytime;                      /*구조  mytime의  주소를  지적자
                                              time_ptr에  대입     */

   total_min = time_in_minutes( &mytime);       /* mytime구조의  주소를  전송       */

    printf("Hours = %d", mytime.hour);       변수에  주소를  전송
    printf("Minutes = %d", mytime.minute);
    printf("Total minutes = %d", total_min);

    total_min = time_in_minutes( time_ptr );     /*구조를  mytime지적자에  전송     */

   printf("Hours = %d", time_prt->hour);
   printf("Minutes = %d", time_prt->minute);
   printf("Total minutes = %d", time_prt->tot_min);
}
/* 분으로  총  시간수를  계산하는  함수           */

int time_in_minutes( struct time *t )  ◄────── 함수소개는  구조를  가리키는
{                                              지적자를  파라메터로  가진다.
 int total_minutes;
 total_minutes = t->hour*60;                 /*시를  분으로  변환        */
 total_minutes +=t->minute;                  /*분을  더한다*/
 return ( total_minutes);
}
```

그림 22-14. 성원변수들의 참조

기타 자료형

C 프로그람에서 소개된 매개 변수는 자료형과 보관클라스를 가지고 있다. 자료형은 변수가 어떻게 기억되는가, 어떤 부류의 자료가 쓰이는가, 변수가 어디에서 리용되는가, 그것이 언제 존재하는가, 얼마나 초기화되였는가에 대하여 결정한다.

C 언어에서는 프로그람작성자가 typedef 자료형에 자료형의 이름을 줄수 있다.

기억클라스

변수의 기억클라스와 관계되는 속성에는 세가지가 있다. 콤파일러는 변수를 기억에 배치할 때 시작값을 가지고 변수를 초기화할수도 있고 그렇게 하지 않을수도 있다. 이 작용은 초기화에 귀착된다. 변수의 존재기간은 프로그람을 창조하는 때부터 실행기간에 변수를 제거할 때까지이다. 변수는 함수범위내에서도 창조될수 있는데 함수가 종결된후에는 제거된다. 셋째 속성인 작용범위는 프로그람작성의 어느 구역에서 변수가 처리되는가에 따라 결정하는것이다.

변수는 전역적으로 처리될수 있다. 이것은 함수바깥에서 정의되는가 혹은 전체 함수에서 허용되는가 하는것이다. 국부적변수는 함수내부에서만 처리될수 있다.

자동기억클라스

자동기억클라스에서 변수들은 그것들이 정의되는 곳에서의 함수에서 국부적변수들로 된다. 이것은 변수의 존재기간과 작용구역이 함수에 의하여 제한된다는것을 의미한다. 다시 말하여 자동기억클라스는 함수가 수행될 때 존재하게 된다. 그 변수는 함수범위내에서만 리용될수 있으며 함수가 끝날 때에는 자동적으로 제거된다. 형태는 다음과 같다.

```
auto  data_type  variable_name;
```

정적기억클라스

정적기억클라스의 변수들은 때때로 전역적으로 고찰되기도 한다. 왜냐하면 이 변수들의 작용구역과 존재기간은 함수에 의하여 제한되기때문이다. 정적변수가 존재하는 기간이 프로그람의 존재기간으로 된다. 변수의 작용구역은 어떻게 정의되는가에 관계된다. 형식은 다음과 같다.

```
static  data_type  variable_name;
```

정적변수를 정의하는 방법에는 세가지가 있다.

- 함수안에서 정의된 정적변수는 그 함수의 정적변수로 제한된다. 함수에서 정의된 정적변수가 아닌 변수는 함수의 수행이 끝난 다음 제거된다. 변수에 값이 주어 지면 값은 여전히 함수가 수행된 다음에도 그 변수에 있게 된다.
- 파일에 있는 모든 함수바깥에서 정의된 정적변수는 그 작용구역으로서 그 원천파일안에서와 모든 함수들에서 효력을 가진다. 정적변수들을 리용하는 목적은 그 변수들을 처리할 때 다른 원천파일에서 사용하는 함수들을 보호하게 하려는데 있다.
- 변수가 함수밖에서 정의되고 정적으로 리용하지 않으면 다른 원천파일에서도 허용되며 외적제한을 리용하여 다른 원천파일의 함수에서도 허용될수 있다. 변수는 매개 원천파일에서 소개하여야 한다. 소개는 우선 변수에 대한 소개를 하는것이고 다음은 외적제한에 대한 소개를 한다.

형정의

typedef 는 프로그람작성자가 자료형에 이름을 붙일수 있도록 한다. 이름으로는 쉽게 구별되도록 대문자를 사용하는것이 관례이다. 실례로 일부 프로그람은 론리적으로 바이트를 사용할수 있다. 프로그람작성자는 unsigned char 로서 바이트로 리용된 모든 변수를 정의하는것을 희망할수 있다. typedef 실마리어를 리용하여 다음의 명령문을 가지고 unsigned char 자료형태를 BYTE 라는 단어로서 쉽게 소개할수 있다.

```
typedef unsigned char BYTE;
```

변수들은 다음과 같이 정의할수 있다.

```
BYTE intial_byte;
BYTE new_byte;
```

원래 BYTE 형으로 정의된 임의의 변수는 실지로 unsigned char 에 의하여 정의된다. 왜냐하면 이것을 조직하고 프로그람을 읽는데서 도움이 되기때문이다.

렬거형

렬거형은 이름에 의하여 련관되는 가능한 값들은 제한된 모임변수들을 만드는 기능이다. 렬거형을 정의할 때의 자료는 이름으로 되는 옹근수상수의 목록이다. 이 상수들은 옹근수가 리용되는 아무곳에서나 리용될수 있다. 렬거형은 다음과 같이 정의할수 있다.

```
enum enum_tyep_name ｛ enumeration list ｝ variable_list;
```

실례에서 변수는 그 주에 대한 날들만 접수할수 있다. 이것은 다음과 같이 정의된다.

```
enum Week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};
```

Week 라고 부르는 렬거형자료들과 이 형변수들은 대괄호({})안에 기입된 값들만 가질수 있다. Week 변수가 다른 값으로 설정되면 오유를 발생시킨다.

Week 의 매 날은 자동적으로 고점옹근수값의 표시로서도 정의된다. 실례로 일요일은 0, 월요일은 1, …등과 같이 정의한다.

아래의 실례에서 yesterday 변수는 Thursday 로 초기값을 가지는 렬거형 week 를 정의한다. 이것에 대한 명령문은 다음과 같다.

```
enum Week yesterday=Thursday;
```

동적기억할당

프로그람작성자는 프로그람안에서 자료를 기억하는데 얼마만한 기억이 필요한가 하는데 대하여서는 알지 못한다. 실례로 프로그람이 문장과 같은 자료를 건반으로 입력할 때 그러하다. 프로그람은 문장을 기억할수 있도록 큰 기억의 부분을 배치할수 있지만 프로그람작성자는 실지로 그 기억이 얼마나 큰지 잘 모른다. 프로그람작성자가 많은 량의 기억을 할당하고 짧은 문장을 기억시키면 기억은 랑비된다. 프로그람작성자가 문장에

100 개 단어만 가진다고 가정하였다면 101 개 단어를 가진 문장을 입력시켜야 할 필요성도 있다. 이것은 프로그람을 수행할 때 기억을 할당하는것이 좋다는것을 보여 준다.

표준서고함수 malloc( )는 프로그람작성자로 하여금 필요한 때마다 필요한 기억만을 할당하게 한다. 기억이 더 필요하지 않으면 해당하게 free( )라는 다른 표준서고함수를 호출하여 기억을 해방할수 있다. 동적기억할당은 프로그람작성자가 기억마당을 조종하게 한다. 지적자들은 동적기억마당에서 기본역할을 한다. 지적자들은 할당된 기억기의 주소를 포함한다. 지적자들은 앞의것들을 정의하지만 지적자가 지적하는 기억은 프로그람이 실행되는 기간에만 유용하다.

기억이 요구될 때 프로그람작성자는 필요한 기억크기를 지정하는 malloc 를 사용할수 있으며 콤퓨터는 지정된 기억기주소를 가진 지적자를 등록한다. 호출은 다음과 같이 malloc( )로 한다.

```
char * ptr;
    ptr=(char *) malloc (size);
```

malloc( )는 요구되는 크기의 기억블로크를 배치하며 블로크에 있는 첫 바이트에 void 지적자에로 귀환한다. malloc( )의 결과는 대입연산자의 왼쪽에 있는 지적자의 형태에 따르게 된다. 실례로 옹근수를 사용하기 위한 블로크를 조성하기 위한 호출은 다음과 같이 한다.

```
int * ptr;
ptr=(int * ) malloc (size);
```

malloc( )함수는 실패하면 NULL 지적자를 귀환한다. NULL 은 void 형으로 정의되는 령값을 가지는 지적자이다. malloc( )의 귀환값은 NULL 값으로 성과적으로 수행되였는가를 검증하게 한다.

```
if (ptr==NULL)
{
printf ("Memory allocation error.\n");
return;
}
```

기억블로크가 더이상 필요하지 않으면 그것은 다음과 같이 free( )함수를 호출하여 조작체계에로 되돌려 보낼수 있다.

```
free (ptr);
```

그러면 기억기는 재리용될수 있다. 프로그람이 종결되면 모든 기억이 자동적으로 해방된다. 프로그람에서 기억을 해방하는것은 편리한 프로그람작성수단이다.

C 프로그람작성부분은 사용자에게 빨리 언어를 리해하게 하며 C++언어를 리해하기 위한 기초로 된다. C 언어부분은 더 구체화하고 이미 고찰한것들을 포함하여 더 많은 자료를 주는 C 프로그람작성법책으로 보충하여야 한다.

제２３장. C++프로그람작성의 기초

프로그람작성언어는 어떤 초보적인 문제에 대해서도 프로그람을 복잡하게 한다. 왜냐하면 복잡한 프로그람을 관리하는 프로젝트가 더 커지기때문이다. 어떤 경우에는 복잡한 프로그람을 관리하는것이 프로그람작성자의 능력의 한도를 초과하기도 한다. 이러한 사정은 복잡한 프로그람을 관리하기 위한 보다 좋은 방법이 있을것을 요구한다. 이런 요구로부터 객체지향프로그람작성이 나오게 되였다. 이미 프로그람작성자들속에 널리 알려진 언어인 C 는 객체지향프로그람작성을 지원하지 않는다. 이런 사정은 C 의 질을 높이도록 재촉하였고 결국 C++를 출현하게 하였다.

C++는 C 를 기초로 하여 만들었는데 프로그람작성자들이 객체지향프로그람작성에서 C 와 류사한 기능을 리용하게 한다. C++가 C 의 높은급의 언어이기때문에 C 프로그람과 C++프로그람은 C++콤파일러로 콤파일할수 있다. 이 능력은 프로그람작성자들이 객체지향프로그람을 작성하는데 노력을 들일 필요가 있다는것을 암시해 준다. C++언어는 그것을 강요하지는 않는다.

"C 언어"부분에서 이미 고찰한 개념과 문법은 여기에서도 직접 응용될수 있다. 그렇지만 C++언어에서는 문법이 단순하고 질이 높으며 객체지향프로그람작성에 대한 새로운 개념을 소개하는데서 C 언어와 차이가 있다. 이 부분에서는 이러한 차이점을 설명하며 이미 기능적으로 C 에 있던 기능을 포함하여 표준화된 C++에서의 새로운 개념과 첨가된 문법을 설명하기로 한다.

C++프로그람작성법에 흥미를 가지는 독자들은 이 부분과 관련되여 C++프로그람작성법에 대하여 충분히 서술된 도서들의 내용으로 보충하여야 한다.

C++기초

C++프로그람은 C 프로그람과 같은 형태로 리용한다. C++프로그람은 머리부파일에서 클라스를 정의하며 원천파일에 포함된다. 이것은 이 부분의 뒤에서 설명하기로 한다.

C++원천파일확장자는 .cpp 이다.

개선된 내용들

이미 지적한것처럼 C++는 일부 문법이 단순하며 C 에서 이미 알려 진 기능들을 개선하고 있다. 여기서는 차이점에 대해서만 고찰할것이다.

설명문

C++는 단순한 한개 행명령문으로 쓴다. 한개 행설명문은 다음과 같이 두개 빗선(//)으로 시작된다.

```
// 한개 행명령문
```

설명문들은 /*로 시작되여 */로 끝나지만 이 기호는 여러개 행의 설명문으로 쓰인다.

I/O 체계

I/O 는 입력과 출력을 의미한다. I/O 연산은 프로그람에서 어떻게 통보를 읽어 들이며 출력을 보내는가 하는것이다. "C 언어"부분에서는 C 의 I/O 에 대하여 두개의 함수 scanf( )(입력)와 printf( )(출력)를 사용하였다.

C++에서는 특수한 I/O 체계를 정의하였다. 그것은 기초자료형보다도 조종을 더 필요로 한다. C++의 I/O 는 객체지향체계이며 프로그람작성자가 만드는 객체를 고려하여 통보를 만들수 있다. C++ I/O 체계에 대한 그 이상의 통보는 이 장의 마지막에 제시한 참고문헌들중에서 어느 하나를 참고하면 된다.

출력명령문

C 에서 printf( )함수는 화면에 통보를 표시하게 한다. C++는 cout 를 리용하는데 화면출력은 다음과 같다.

```
cout << "Let' s go to the beach\n";
```

우의 명령문은 삽입자(<<)를 리용하여 화면에 기호렬 "Let' s go to the beach"를 출력한다.

변수를 아래와 같이 화면에 쉽게 표시할수 있다.

```
int num_rooms=4;
cout << "My new house has "<<num_rooms<<"bedrooms.";
```

프로그람에 의하여 계산을 할 때 반드시 다음의 처리지시자를 포함시켜야 한다.

```
#include <iostream>
using namespace std;
```

name space 를 리용하는 자료를 더 보려면 이 장의 마지막에 제시된 "이름공간"부분을 참고하기 바란다.

사용자의 콤파일러가 이전 형인 경우에는 새로운 형의 머리부를 지원하지 않을수 있다. 두개 명령문들은 이전콤파일러에서 제시된 다음의 사항으로 교체해야 한다.

```
#include <iostream.h>
```

입력명령문

C 에서는 사용자가 함수 scanf( )를 리용하여 건반으로부터 입력한 통보를 읽는다. C++에서는 cin 이 있는데 이것은 콘솔입력이다. 이것은 아래와 같이 건반에서만 입력한다.

```
int  length;
cout << "Enter the length of the vehicle:";
cin >> length;
```

우의 코드토막은 사용자에게 입력시킬것을 재촉하는데 그러면 사용자는 닫기연산자(>>)에 의하여 건반으로 자료를 입력시킨다.

cin 을 사용할 때 아래에서 제시된 처리지적자를 포함하여야 한다.

```
#include  <iostream>  using  namespace  std;
```

이름공간에 대한 더 많은 자료를 보려면 "이름공간"부분을 참고하기 바란다.

콤파일러가 이전 형이면 새로운 형의 머리부를 지원하지 않을수 있다. 이전 콤파일러에서 쓰이는 두개 명령문은 다음의 것으로 교체하여야 한다.

```
#include  <iostream.h>
```

머리부

C++ 콤파일러의 이전 방안들에서는 ＃ include 에서 머리부파일에 확장자가 .h 인 파일을 리용하였다. iostream 머리부파일에 대한 낡은 ＃ include 명령문형태의 실례는 다음과 같다.

```
#include  <iostream.h>
```

이것은 전처리기가 프로그람에 iostream.h 파일을 포함하게 한다. 낡은 형태의 지적에서는 파일이름을 사용한다. 이것은 더 새로운 C++표준경우가 아니다.

새로운 C++표준은 머리부파일에 파일이름을 지적하지 않고 표준식별자를 쓴다. 머리부는 프로그람에서 요구되는 특수한 통보를 삽입하게 한다. 식별자는 파일이름일수도 있고 그렇지 않을수도 있다. 이에 대한 실례는 다음과 같다.

```
#include  <iostream>
```

새로운 C++머리부형을 포함할 때 머리부안의 내용은 std 이름공간에 포함된다. 이름공간에 대한 더 많은 통보를 보려면 이 장의 마지막에 있는 "이름공간"을 참고하기 바란다.

렬거형

C++에서 Week라는 렬거형변수를 소개할 때 다음과 같은 명령문을 쓴다.

```
enum Week yesterday;
```

또는 다음과 같이 쓸수도 있다.

```
Week yesterday;
```

우선권

몇가지 새로운 연산자에는 우선권을 포함시켜야 한다. 풀이연산자(::)는 가장 높은 우선권을 가진다. 이 부분자료는 이 장의 마지막에 제시된 "이름공간"을 보시오. C++는 4개의 새로운 형타연산자를 포함한다. 이 연산자에 대한 더 많은 통보를 보려면 이 장의 마지막에 제시된 "형타연산자들"을 참고하기 바란다.

표 23-1은 우선권순위로 연산자를 가장 높은 순서로 기입한다.

표 23-1 우선권순서로 된 C++연산자들

연 산 자	련 관 성
::	왼쪽에서 오른쪽으로
( )	왼쪽에서 오른쪽으로
한 인수+, 한 인수-, sizeof( ), ++, --, (type)	오른쪽에서 왼쪽으로
static_cast, dynamic_cast, const_cast	
reintrept_cast	
*, /, %	왼쪽에서 오른쪽으로
두 인수+, 두 인수-	왼쪽에서 오른쪽으로
=, +=, -=, *=, /=, %=	오른쪽에서 왼쪽으로

C++의 새로운 특성

다음의 특성들은 C++의 표준부분이며 C에서는 그것에 대한 토대도 가지고 있지 않다. 이 많은 특성들은 C++의 기능을 강화해 주며 객체지향프로그람작성을 가능하게 한다.

이름공간

이름공간은 최근에 C++표준에 첨가된것이다. 이러한것으로 해서 이름공간은 C++콤파일러에서 지원을 받지 않는다.

이름공간들은 이름들이 겹치는것을 피하도록 설계되여 있다. 이름공간앞에는 이 모든 이름들이 전역적이름공간에서 **슬로트**(slot)들에 대하여 초점을 이룬다. 전역적이름공간이나 전역적범위는 모든 함수나 클라스들의 바깥에서 같은 지역으로 생각할수 있다. C 혹은 C++에서 함수바깥에 있는 원천파일의 꼭대기에서 정의된 임의의 변수는 전역적이다. 이런것은 그 변수들이 모든 함수들에서 허용되게 한다. 대역이름공간에서 이름들에 대한 고찰은 특히 C++프로그람작성환경에서 심화되기때문에 여기에는 변수이름, 함수 그리고 클라스 이름들이 더 많다.

실례로 표준서고에는 abs( )함수가 들어 있는데 이 함수는 옹근수인 절대값을 준다. 프로그람에서 abs( )라고 하는 함수로만 정의하면 표준서고함수 abs( )를 가지고 전역적 이름공간에서 겹친다. 프로그람작성자는 표준서고에서 찾지 못하는 함수이름을 리용하는데서 조심한다하더라도 세개의 부분서고와 겹칠수 있다. 이것은 함수, 변수, 클라스이름을 유일하게 한다.

이름공간은 이름자체에 값을 주는 프로그람구역이며 그 구역에서 정의되는 모든 이름들은 그 이름공간에서 국부화된것이다. 이름공간에서 정의된 임의의 이름들은 그 이름과 관련된 이름공간의 이름을 가진다. 그림 23-1은 두개의 이름공간에 대하여 설명한다.

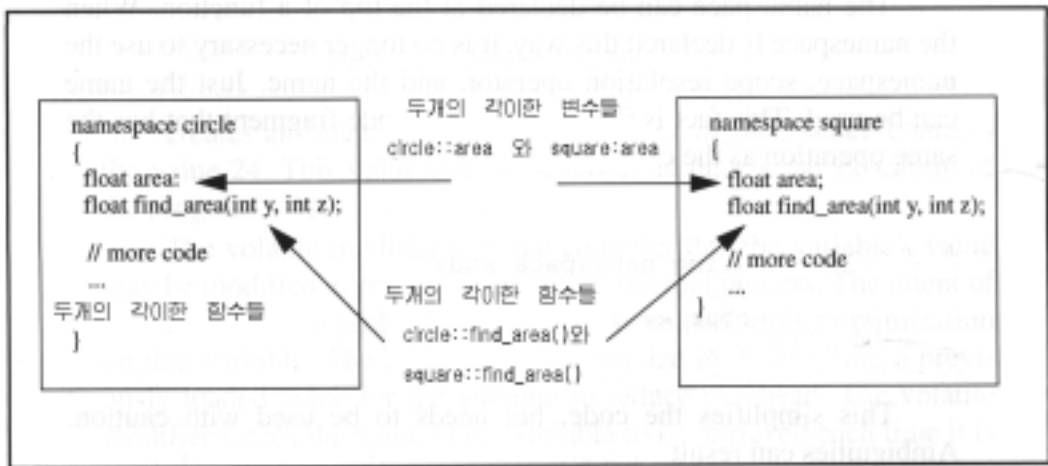



그림 23-1. 두개의 이름공간들

매개 이름공간에는 area 라고 부르는 변수가 있다. 매개 변수는 개별적인 이름공간과 련관되여 있다. 그것들은 꼭 같은것으로 보이나 실지 변수이름들은 circle::area 와 square::area 이다. 동일하게 함수 circle::find_area( )와 square::find_area( )를 true 로 취한다. 이런 이름을 붙이는것은 그것들사이의 임의의 충돌을 방지한다.

표준서고이름공간은 std 이다. 표준서고에서 정의된 임의의 이름은 그 이름과 관련된 std 이름을 가진다. 실례로 콘솔출력명령문 cout 의 완전한 이름은 std::cout 이다. 변수내용을 표시하기 위한 명령문을 다음과 같이 제시할수 있다.

```
std::cout << value1;
```

연산자 ::는 령역해결연산자이다. 이 연산자는 콤파일러가 어디에서 이름의 정의를 찾아야 하는가를 알려 준다. 웃 실례에서 콤파일러는 표준서고에서 cout 정의를 찾는다.

using 지령은 이름공간에 있는 모든 이름들이 다음 단계에로 넘어 가지 않고 리용되게 할수 있다. using 지령의 형식은 다음과 같다.

```
using namespace namespace_name;
```

이름공간은 함수의 맨 우에서 정의될수 있다. 이름공간이 어떤 방식으로 소개되면 더는 이름공간, 령역해결연산자, 이름을 사용하는것이 필요하지 않다. 이 사실은 우의 실례에서와 같은 연산을 가진 코드토막에 의하여 설명할수 있다.

```
…
using namespace std;
…
cout << value1;
…
```

이것은 코드를 단순하게 하지만 모호성이 생길수 있으므로 주의하여 리용하여야 한다. 코드를 단순화하기 위해서는 using 지령대신에 using 선언을 리용하는것이 더 좋다. using 선언의 형식은 다음과 같다.

```
using namespace_name::identifier;
```

만일 어느 한 프로그람이 cout 를 자주 리용한다면 using 선언을 다음과 같이 리용할수 있다.

```
using std::cout;
```

기타 자료형과 연산자

C++은 몇가지 다른 자료형들과 연산자를 지원한다. 또한 C++의 동적기억할당도 론의하게 한다.

처리변경자

변수가 처리되거나 변경되는 방식을 조종하는데 리용되는 처리변경자에는 두가지가 있다. 첫째의것은 C 언어에도 있는 const 이다. 둘째의것은 C++에만 있는 volatile 이다. 변수가 const 변경자로 정의될 때 프로그람수행기간에는 그 값들을 변경할수 없다. 실례로

```
const int hours_in_day = 24;
```

는 hours_in_day 라고 부르는 옹근수변수를 만들면서 값 24 를 준다. 이 값은 프로그람의 수행기간에만 변경될수 없다.

변경자는 콤파일러에 변수값이 외부처리에 의하여 변경되지 말아야 한다는것을 알려준다. 이 변경자의 목적은 콤파일러가 그 변수에 관한 최량화를 수행하지 못하게 하는데 있다. 콤파일러는 전체적으로 작업량을 감소시키기 위하여 이미전에 변수에 적재한 값을 련결시켜 생략할수 있다.

재설정변경자(volatile)는 변수값이 리용될 때마다 변수값이 설정되게 한다.

실례로 time 이라는 변수에 의하여 콤퓨터에 설정된 시계장치의 매 초로 변경시킬 필요가 있다면 콤파일러는 이전에 설정된 값을 다시 리용하지 말아야 한다. 이때의 변수 time 은 다음과 같이 정의한다.

```
volatile  int  time;
```

이것은 변수 time 이 그 변수가 련결될 때마다 새 값으로 설정된다는것을 보여 준다.

동적기억할당

C++는 동적기억할당을 위한 두개의 새로운 연산자를 가지고 있다. 그것은 New 와 Delete 이다. New 는 기억을 할당하며 Delete 는 New 에 의하여 할당된것을 없앤다. 이것의 형식은 다음과 같다.

```
pointer_variable=new data_type;
delete pointer_variable;
```

pointer_variable 은 data_type 형의 지적자이다. New 연산자는 data_type 형의 값을 취하는데 충분한 기억을 할당하며 그 기억에 대한 지적자를 귀환시킨다.

Delete 연산자는 pointer_variable 에 할당된 기억을 해방시킨다. 기억은 New 연산자를 사용할 때 자료형뒤의 괄호안에 쓴 값에 의하여 초기화한다.

실례 :

```
#include <iostream>
 using namespace std;

int main( )
{
int *num_ptr;
          num_ptr = new int (346); //346 으로 초기값주기
          if (!num_ptr)
          {
cout << "Memory allocation failure.\n");
return 1;
          }

          cout << "Memory was successfully
```

```
                allocated for num_ptr containing " << *p;

        delete   p;           //기억지우기
        return   0;
    }
```

배렬들도 New를 사용하여 할당할수 있다.
1차원배렬을 할당하려면 다음과 같이 한다.

```
pointer_variable=new  variable_type[size];
```

크기는 배렬에 있는 원소수에 관계된다.
할당된 배렬을 해방하려면 다음과 같이 한다.

```
delete[  ]pointer_variable;
```

함수의 겹쳐넣기

함수의 **겹쳐넣기**(Overloading)는 두개이상의 함수들이 같은 이름을 가지며 서로 다른 파라메터를 가지는 함수를 정의할수 있게 하는 C++의 특성이다. 함수들이 같은 이름을 가질 때 그것들은 **겹쳐넣기되였다**고 말한다. 함수의 겹쳐넣기는 C++가 다형성을 형성하는 한가지 방법이다.

다음의 프로그람을 고찰하자.

```
// 3개까지 함수를 겹쳐넣기
# include <iostream>
using namespace std;
void print_values( int x );          //함수원형-옹근수파라메터
void print_values( int x, int y);    //두개옹근수파라메터들
void print-values(double z );        //배정확도파라메터
int main ( )
{
print_values(346);
print_values(3, 346);
print_values(10.25);
return 0;
}

void print_values(int x)
{
```

```
            cout << "The value is"     << x;
}
void print_values(int x, int y)
{
            cout << "The value is"     << x <<"and"<< y;
}

void print_values(double z)
{
cout << "The value is " << z;
}
```

print_values( )함수는 함수가 3 가지 서로 다른 파라메터목록을 선택하면서 3 번 적재된다.

함수의 겹쳐넣기는 린접한 련관연산을 위한것이다. 실례로 square( )라고 부르는 함수를 겹쳐넣기하면 처음에는 옹근수 2 차뿌리, 다음에는 류동점수의 2 차뿌리를 계산하는데 이것은 우월한 프로그람작성수법이 아니다. 겹쳐넣기함수에 대한 변형은 거의나 련관되여 있지 않다. square( )함수가 옹근수를 접수하고 그것의 2 차뿌리를 계산하고 또 류동점수를 접수하도록 함수를 겹쳐넣기하고 2 차뿌리를 계산하면 연산이 린접한것으로 련관되고 겹쳐넣기가 은을 내게 된다.

기정적인 함수인수

C++에는 대응하는 값이 없이 호출되면 기정값을 가질수 있는 파라메터가 있다. 실례로 함수가 옹근수와 파라메터목록에 있는 문자로 호출되면 이 매개 값은 다음실례에서와 같이 기정값을 받을수 있다.

```
void  price_widgets(int num_widgets=1, char widget_type='A'
{
            //처리코드부분
…
}
```

함수는 그때 다음과 같은 방식으로 호출된다.

```
price_widgets(3, 'D');      //값들이 모든 파라메터에 들어 간다.
price_widgets(7);           // widgets 에 수가 들어 간다.
                            //그러나 기정값이 사용된다.
                            // widgets 형에 대하여
price_widgets( );           //파라메터들이 동시에 기정값들이 사용된다.
```

첫째 함수호출에서는 값들이 모든 파라메터에 들어 간다. 둘째 함수호출에서는 widget_type 가 값을 가지지 않으면 'A'가 기정으로 리용된다. 셋째 함수호출에서는 두개 파라메터가 들어 가지 않는다. widget 의 수자들에 대한 기정값 1 과 widget 형태는 'A'가 리용된다.

클라스

클라스는 객체지향프로그람작성에서 C++지원의 기초로 된다. 그 클라스는 프로그람이 객체라는 용어를 쓰도록 한다.

실마리어클라스는 새로운 자료형을 정의하는데 리용되며 자료형은 객체를 만드는데 리용된다. 클라스는 자료를 조작하는 자료와 코드를 정의하고 포함한다. 객체의 범위내에서 자료로 연산하는 자료성원들과 함수들을 함께 묶어 요약한다.

클라스가 정의될 때 객체가 작성된다. 이 객체는 기억기에서 기억공간을 배치하는 물리적구체례이며 클라스의 실례이다. 클라스의 매 객체는 클라스에서 정의된 자료의 복사를 가지고 있다. 클라스정의형태는 다음과 같다.

```
class  class_name{
                private data and functions
public：
                public data and functions
}object_list;
```

개별적인 자료와 함수들은 클라스에서 함수들에 의해서만 리용될수 있다. 임의의 개별적인 자료나 함수들은 외부의 간섭으로부터 보호된다. 이 보호는 자료은폐로서 잘 알려 져 있다.

프로그람에 의하여 은행의 화페자료를 관리할 때 계산서내에서의 자료를 직접 처리하려고 하거나 계산서의 균형을 변화시키는 프로그람을 외적요인으로부터 보호할 필요가 있다고 하자. 클라스는 개별적으로 계산서의 균형을 정의하여 보호한다.

클라스에 대한 실례는 다음과 같다.

```
// 이것은 bankaccount 클라스를 창조한다.

class  bank_acct {
        double acct_balance;
        public：
                void init( );
                double retrieve_balance( );
                void update_balance( );
                …

};
```

기정적으로 클라스안의 모든 항목들은 독자성을 띤다. 이 클라스에서 acct_balance 는 개별적으로 다른 bank_acct 클라스의 다른 성원들에 의하여서만 처리될수 있다. 대역지적자다음에 정의된 모든 항목들은 프로그람안에서 모든 다른 함수들에 의하여 처리될수 있다. 대표적으로 프로그람은 한개의 클라스안에서 그 프로그람의 대역함수를 거쳐 개별적인 항목들을 처리한다. 함수 init( ), retrieve_balance( ), update_balance( )는 성원함수들이라고 부르는데 대역대면부를 만든다. 성원함수들의 원형이 클라스의 정의내부에 있으므로 그것들은 아무곳에서나 원형으로 될것을 요구하지 않는다.

클라스가 정의된 후에 클라스이름을 리용하면 객체는 그런 형으로 창조될수 있으며 클라스이름은 새로운 자료형지적자로 된다. 검사와 기억의 두개 객체를 만들려면 다음 명령문을 사용하면 된다.

```
bank_acct  checking, savings;
```

retrieve_balance( )와 같은 성원함수를 실행시키려면 콤파일러는 어떤 부류의 함수가 함수이름을 지정하여 어느 클라스에 소속시키는가를 알아야 한다. retrieve_balance( )함수의 원천코드는 다음과 같다.

```
double  bank_acct::retrieve_balance( )
{
        return(acct_balance);
}
```

기호 ::는 풀이연산자이다. 이 연산자는 함수의 retrieve_balance( )가 bank_acct 클라스에 속한다는것을 콤파일러에 전달해 준다.

클라스의 부분이 아닌 프로그람내부에서 성원함수를 호출하려면 객체이름과 점연산자를 리용하여야 한다.

다음의 실례는 객체검사를 위한 retrieve_balance( )를 호출하는것이다.

```
bank_acct checking, savings;
double checking_balance;

checking_balance=checking.retrieve_balance;
```

이것은 객체검사만을 위한 계산균형을 맞춘다.

구축자

객체의 일부 부분이 초기화를 요구하는것이 보통이다. bank_acct 클라스에서 acct_balance 는 0.0 으로 설정되여야 한다. 이것은 구축자함수로 수행된다. 이 함수가 제공되지 않으면 콤파일러는 기정적으로 구축자를 제공한다. 클라스의 객체는 구축자에 의

하여 만들어 지는데 이것은 기정적인 구축자가 할수 있는 모든 기능들이다.

구축자함수는 클라스의 성원이며 클라스와 같은 이름을 가진 특수한 함수이다. 객체가 작성될 때에는 객체의 구축자가 호출된다. 클라스를 위한 구축자함수 bank_acct 의 실례는 다음과 같다.

```
//구축자함수
bank_acct::bank_acct( )
{
        acct_balance=0.0;
}
```

전역적인 객체에 대하여 구축자는 main( )함수앞에서 호출되며 부분적인 객체에 대하여 객체가 우연히 나타날 때까지 호출된다.

구축자도 파라메터를 접수할수 있다.

이것은 다음과 같이 볼수 있다.

```
// 이것은 bank account 클라스를 창조한다.
class bank_acct {
        double acct_balance;
public：
        bank_acct(double starting_balance);
                        //구축자파라메터들을 받는다.
        ~bank_acct( );    //파괴자
        void init();
        double retrieve balance( );
        void update_balance( );
        …
};
//구축자함수
bank_acct::bank_acct( double balance)
{
                acct_balance = balance;
}
```

객체가 다음과 같은 방법으로 정의될 때 구축자함수인수가 통과된다.

```
bank_acct  checking=bank_acct（120.05）;
```

혹은

```
bank_acct  checking（120.05）;
```

파괴자

이 함수는 객체가 파괴될 때 호출된다.

이 함수는 클라스의 성원이고 클라스와 같은 이름을 가지지만 파괴자의 이름앞에는 ~가 붙는다. bank_acct에 대한 이 함수의 실례는 다음과 같다.

```
//파괴자함수
bank_acct::~bank_acct( )
{
        //일지파일에 대한 통보를 계수하여 쓴다.
        …
}
```

함수에 대한 원천코드가 클라스범위내에 있으면 클라스이름과 풀이연산자가 필요되지 않는다.

동료함수

한 클라스의 성원은 국부적이나 전역적일수 있다. 국부적인 성원들은 클라스안의 다른 성원들만이 개별적인 임의의 국부적인것을 리용할수 있다. 전역적인 성원들은 그것들이 정의된 클라스범위함수밖에 있는 클라스들과 함수들에 의하여 처리될수 있다.

클라스의 성원은 아니니지만 그 전체 성원들을 처리할수 있는 함수를 **동료함수**(Friend Function)라고 부른다. 련관함수는 비성원함수들이 국부적인 클라스들을 처리할수 있게 한다. 클라스의 대역부분에서는 friend라는 실마리어를 리용하고 함수원형을 포함시켜야 한다. 실례로

```
//이것은 클라스 bank account를 창조한다.

class bank_acct {
        double acct_balance;
public:
        bank_acct(double starting_balance);
                //구축자가 파라메터를 받는다.
        ~bank_acct( );              //파괴자
        void init( );
        double retrieve_balance( );
        void update_balance( );
        friend void irs_audit(bank_acct x);
            //irs_audit 함수는 클라스부분이 아니다.
            //그러나 국부적성원을 처리할수 있다.
        …
};
```

```
//irs_audit 함수는 임의의 클라스의 성원함수가 아니라는것을 주의할것.
 void irs_audit( bank_acct x)
{
            //코드부분
            …
}
```

irs_audit( )함수는 bank_acct 클라스의 전체 성원들을 직접 처리할수 있다. 실례로 irs_audit( )함수는 다음과 같이 하여 bank_acct 에 있는 acct_balance 를 처리할수 있다.

```
x.acct_balance;
```

기호렬객체

C++는 기호렬을 제공하는데 이것은 클라스에 의하여 정의된다. 클라스는 언어에서 새로운 형만을 소개한다. 정의된 형으로 클라스를 리용하는것은 기호적인 자료형들중에서 하나를 요구하는것과는 다르다.

C 에서 리용된것과 마찬가지로 정확하지 못한 기호렬들은 임의의 C++연산자에 의하여 처리될수 없다. 표준기호렬클라스는 간편할뿐아니라 배렬경계의 초과를 방지하게 한다.

기호렬객체의 정의

기호렬객체는 char 형기호들의 기호렬을 포함한다. 빈 기호렬과 같은 기호렬객체를 정의하려면 다음의 명령문을 리용한다.

```
string student_name;
```

기호렬을 정의하고 초기화하려면 다음의것을 리용하든지

```
string student_name="Glenn";
```

또는 아래에서처럼 함수정의를 리용하여야 한다.

```
string student_name("Glenn")
```

보관된 기호렬은 령끝기호를 필요로 하지 않으며 기호렬객체는 기호렬길이의 자리길을 보존한다.

기호렬대입

기호렬은 대입연산자를 리용하여 문자 혹은 다른 기호렬의 값을 준다.

```
string  valedictorian = "Carly";         //기호렬정의와 초기화
string student_1 = "Liz";                //기호렬정의와 초기화
valedictorian  =  student_1;             // valedictorian의 내용변경
```

련결

기호렬은 +연산자를 리용하여 서로 련결된다. 다음의 실례는 련결을 보여 준다.

```
string valedictorian = "Carly"; //기호렬정의와 초기화
string student_1 = "Liz";             //기호렬정의와 초기화

string announcement = valedictorian + " and " + student_1 + " have won awards."
```

비교

전체 기호렬을 비교할 때 임의의 비교연산자를 가지고 기호렬객체를 리용한다. 아래에 비교연산자들을 제시한다.

```
>,   >=,   <,   <=,   = =,   !=
```

기호렬의 비교실례는 다음과 같다.

```
string animal_1 = "dog";
string animal_2 = "cat";

if (animal_1 = = animal_ 2)
          cout << "Both animals are the same.\n");
else
          cout << "These animals are different.\n");
```

계승성

계승은 이미 있던 클라스를 다시 리용하거나 확장시켜 새 클라스들이 창조되게 한다. C++는 다른 클라스를 하나의 클라스로 정의하여 결합하여 계승을 제공한다.

토대클라스는 그림 23-2 에서 보여 주는바와 같이 새로운 클라스 또는 유도클라스를 작성한다.

계승형태는 다음과 같다.

```
class derived_class : access base_class
{
   body of new class
}
```

처리는 선택적으로 할수 있지만 만일 선택한다면 public, private 또는 protected 를 선택할수 있다.

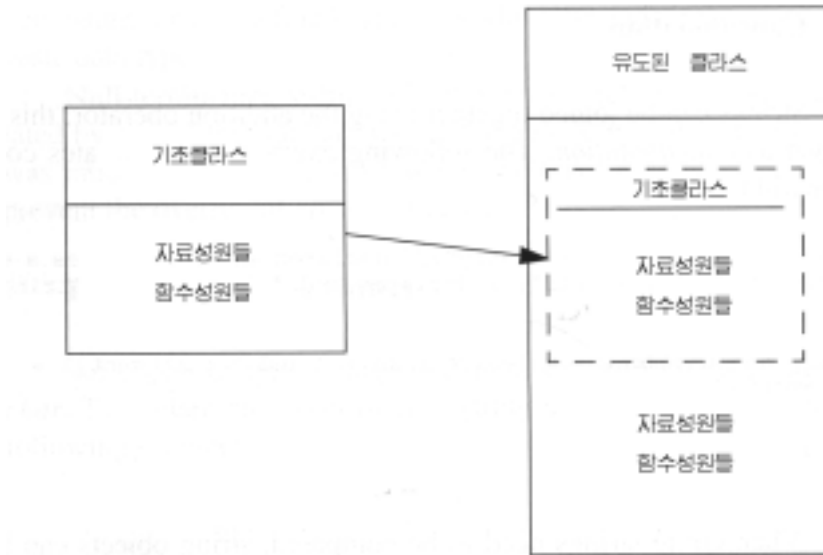



그림 23-2. 토대클라스가 새로운 유도클라스를 만든다.

다음의 클라스 automobile 는 승용차에 대한 해설이다. 문의 개수, 바퀴의 개수, 기관에서 기통의 개수를 준다.

```
class automobile
{
        int doors;
        int wheels;
        int cylinders;
public:
        void set_doors(int num_d) {door = num_d ; }
        int get_doors( ) { return doors ; }
        void set_wheels(int num_w) {wheels = num_w ; }
        int get_wheels( ) { return wheels ; }
        void set_cylinders(int num_c) {cylinders = num_c ; }
        int get_cylinders( ) { return cylinders ; }
};
```

automobile 의 토대정의는 지정된 객체를 정의하는데 리용된다. 실례로 automobile 이라는 토대클라스는 station_wagon 이라는 클라스를 정의하는데 리용된다.

```
class station_wagon  : public automobile
{
        int hatch_space ;
public:
        void set_hatch( int size) {hatch_space = size ; }
```

```
        int get_hatch( ) { return hatch_space ; }
        void display_features();
};
```

station_wagon 이 automobile 을 계승하기때문에 station_wagon 클라스는 automobile 클라스의 모든 성원을 포함하며 승용차의 뒤에 있는 짐차용적에 대한 자료와 성원함수를 보충한다.

접근조종

한 클라스가 다른 클라스에서 계승될 때 토대함수성원들은 유도클라스의 성원으로 된다. 이것은 유도클라스가 클라스이기만 하면 성립된다. 유도클라스는 구조일수 있으며 그 경우에 기정적인 접근은 public 이다. 접근지적자로 토대함수의 성원들을 접근하는데는 유도클라스의 기능에 관계된다.

우선 클라스의 개별적인 성원들은 public, private 또는 protected 로 정의된다. 클라스의 Private 성원은 클라스의 성원함수에 의해서만 접근할수 있다. 금지된 성원함수는 계승될 때만 유도클라스의 접근을 진행하며 그렇지 않으면 클라스의 개별적인 성원처럼 행동한다. 계승은 토대클라스의 성원함수접근에 영향을 준다. 그리고 토대함수는 그것과 련관된 접근지적자를 가질수 있다. 클라스가 유도되거나 토대클라스에서 계승되면 public, private, protected 토대클라스의 접근지적자가 선택된다. 토대클라스함수성원들의 조밀성은 토대클라스처리와 성원함수접근의 총 9 개 결합에서 설명될수 있다.

- 토대함수의 계승이 public 일 때(계승된 성원함수들의 접근상태는 변하지 않는다.)
  1 전역적으로 계승된 성원함수들은 대역이다.
  2 보호로 계승된 성원함수는 보호된다.
  3 국부로 계승된 성원함수는 국부이다.

- 토대클라스의 계승이 private 일 때
  4 대역으로 계승된 토대클라스성원함수들은 유도클라스에서 보호된 성원함수들로 된다.
  5 보호로 계승된 토대클라스의 성원함수들은 보호로 남아 있다.
  6 국부적으로 계승된 토대클라스의 성원함수들은 국부로 남아 있다.

- 토대클라스의 계승이 protected 일 때
  7 전역적으로 계승된 토대클라스의 성원함수들은 유도클라스에 대하여 국부로 된다.
  8 보호된 토대클라스의 성원함수들은 유도클라스에 대하여 국부로 된다. 그것들은 클라스의 성원함수들에 의하여 처리될수 있으나 그외의 유도클라스에 의해서는 처리될수 없으며 지어 다른 유도클라스가 계승될 때에도 처리될수 없다.
  9 토대클라스의 국부성원함수들은 유도클라스에서 국부로 남아 있다.

다형성

여기서는 C++에서 개선된 다형성에 대하여 설명한다. C++다형성은 함수의 서로 다른 함수수행을 같은 이름으로 처리될수 있는가 하는것인데 "한개 대면부, 다형성"이다. 다형성은 클라스계층범위에서만 움직이며 하나의 클라스를 유도하는 능력을 가진다. C++다형성은 번역편집할 때와 실행할 때에 제공된다. 함수초과적재는 번역편집할 때의 다형성실례이다.

C++은 또한 매개 유도클라스당, 가상함수당, 실행할 때의 다형성을 허용한다.

토대클라스지적자

실행할 때 다형성의 토대는 토대클라스지적자이다. 토대클라스와 유도클라스의 지적자들은 다른 지적자들과는 다른 특수한 관계를 가지고 있다. 이미 한개형태의 지적자가 다른 형태의 객체로 지정될수 없다는것은 보았다. 토대클라스지적자와 객체들은 규칙에서 례외이다. 토대클라스지적자는 2 진토대에서 유도된 임의의 클라스의 객체를 지정하는데 리용된다. 실례로 자동차의 토대클라스와 station_wagon 의 유도클라스가 주어 져 있을 때 자동차의 지적자로 정의된 임의의 지적자는 station_wagon 지적자일수 있다.

```
class automobile
{
        int doors;
        int wheels;
        int cylinders;
public:
        void set_doors( int num_d ) {door = num_d ; }
        int get_doors( ) { return doors ; }
        void set_wheels( int num_w ) {wheels = num_w ; }
        int get_wheels( ) { return wheels ; }
        void set_cylinders( int num_c ) {cylinders = num_c ; }
        int get_cylinders( ) { return cylinders ; }
};

class station_wagon : public automobile
{
        int hatch_space;
public:
        void set_hatch(int size) {hatch_space = size ; } int
```

```
get_hatch( ) { return hatch_space ; }
          void display_features( );
};
```

다음에 주는것 :

```
automobile P*;              //automobile형의 객체에 대한 지적자
automobile auto_object;     //automobile station_wagon에 대한 객체
stwgn_object;               //station_wagon형에 대한 객체
```

아래 두개 명령문이 효과적이다.

```
    p = &auto_object;       //automobile형에 대한 객체를 지적하는 지적자
    P = &station_wagon;     //station_wagon형에 대한 객체를 지적하는 지적자
```

점 p 는 automobile 로부터 넘겨 받은 station_wagon 의 모든 요소를 처리할수 있다. 그러나 형이 사용되지 않는 한 station_wagon 형의 구제적인 요소들은 p와 함께 련결될수 없다.

가상함수

다형성은 가상함수와 계승의 결합을 통하여 수행된다.

가상함수는 토대클라스에서 가상적으로 소개되며 하나 또는 그이상의 유도클라스들 속에서 재정의된다. 이 정의는 매 유도클라스들이 가지고 있는데 가상함수방안을 가지게 한다. 가상함수는 토대클라스지적자를 통하여 호출된다. C++은 실행기간에 호출을 위한 어느 가상함수의 방안이 이 지적자에 의하여 지적된 객체의 형에 토대하고 있는가를 결정한다. 콤파일러는 토대클라스로부터 유도된 임의의 클라스에서 동적으로 묶는다. 지적된 객체형들은 실행하기 위한 가상함수방안을 결정한다.

다음프로그람은 가상함수의 사용에 대하여 보여 주고 있다.

```
#include <iostreadm>
using namespace std;
class base_print
{
public:
          virtual void print_m( )
          {
          cout << "Print statement from the base class.\n";
          }
};
```

```
class first_print:public base_print
{
public:
   void print_me( ) {cout << "Print statement from first_print class.\n";}
};

class second_print:public base_print
{
public:
   void print_me( ) {cout << "Print statement from second_print class.\n"}
};

int main( )
{
          base_print base_object;
          base_print ptr;
          first_print first_object;
          second_print second_object;

          p = & base_object;
          p->print_me( );          // base_print 클라스출력

          p = &first_object;
          p->print_me( );          //first_object 클라스출력

          p = &second_object;
          p->print_me( );          //second_object 클라스출력

          return 0;
}
```

이 프로그람은 다음과 같은 결과를 인쇄한다.

```
print statement from the base class.
print statement from the first_print class.
print statement from the second_print class.
```

형변환연산자

C 언어부분에서 본 캐스트에 보충적으로 C++는 4 개의 캐스트연산자 즉 dynamic_cast, const_cast, reinterpret_cast, static_cast 를 정의한다. 이 연산자들은 const 가 어떻게 실행되는가 하는 조종체에 추가된다.

dynamic_cast

dynamic_cast 는 실행되는 기간 cast 가 유효한가를 검증한다. dynamic_cast 의 목적은 다형성에서 cast 를 실행하기 위한것이다. 원래의 dynamic_cast 는 cast 인 지적자 또는 참조가 목적형의 객체나 목적형으로부터 유도된 객체를 위한 지적자 또는 참조가 목적형의 객체나 목적형으로부터 유도된 객체를 위한 지적 또는 참조이면 계승된다.

dynamic_cast 의 형식은 다음과 같다.

dynamic_cast <target_type> (expression)

const_cast

const_cast 는 cast 에서 고정 또는 변경을 무시한다.

const_cast 의 형식은 다음과 같다.

const_cast <target_type>(expression)

static_cast

static_cast 는 cast 에서 다형성이 아닌 cast 를 실행한다. 이것은 "C 언어"부분에서 준비된 초기 cast 연산자에 대입된다.

static_cast <data type>(expression)

reinterpret_cast

reinterpret_cast 연산자는 한가지 형을 여러가지 형으로 전환한다. 실례로 그것은 지적자를 옹근수와 다른것으로 바꿀수 있다. 이 경우 모순적인 지적자형이 사용될수 있다.

reinterpret_cast 형식은 다음과 같다.

reinterpret_cast <data_type>(expression)

례외처리

례외는 실행할 때에 일어 나는 오유이다. C++는 프로그람이 조종방법으로 실행될 때 례외로 조종하게 한다. 오유 또는 례외들은 요유조종루틴으로 쉽게 포착되여 조종될 수 있다.

례외조종을 위한 3 개 예약어 즉 try, catch, throw 가 있다. 모니터에 대한 프로그람 작성명령문들은 try 블로크에 포함된다. 례외가 try 블로크속에서 일어 난다면 throw 를 사용하여 통보한다. throw 는 생겨 난 오유에 기초한 례외를 발생시킨다. throw 는 try 블로크 또는 try 블로크기능으로 실행된다. throw 명령문의 형식은 다음과 같다.

```
throw  exception;
```

try 와 catch 의 일반형식은 다음과 같다.

```
try
{
        //try 블로크에 대한 오유검사
        //명령문이나 함수들을 포함
        //블로크부분
}
catch  (type1  arg)
{
        //catch 블로크
}
catch  (type2  arg)
{
        //catch 블로크
}
        …
catch  (typeN  arg)
{
        //catch 블로크
}
```

C++프로그람은 사용자들이 빨리 리해하도록 하기 위하여 리용된다.

제24장. 인터네트프로그람작성기초

인터네트프로그람작성은 많은 방법으로 정의된다. 이 장에서는 인터네트란 무엇인가 하는것과 인터네트의 력사, 일부 프로그람작성방법들에 대해서 고찰한다.

인터네트의 개념

인터네트는 세계적인 큰 콤퓨터망이며 사람들과 정보처리를 진행한다. 이 망은 단순한 망이 아니며 대단히 넓은 범위의 망이다. 인터네트를 통하여 우리는 서로 멀리 떨어져 있어도 우편물을 주고받을수 있으며 음악을 듣고 비데오를 볼수 있다. 또한 독자적인 Web페지를 창조할수 있으며 다른 Web페지들도 처리할수 있다.

인터네트는 여러가지 봉사처리를 진행한다. 인터네트는 부단히 발전하고 있으며 인터네트에 가입하는 사람들은 날을 따라 늘어 나고 있다. 인터네트는 새로운 사용과 기술의 동력으로 되며 하나의 정보혁명이라고 말할수 있다.

력사

1960년대 초에 전화방송망을 거쳐 콤퓨터들사이를 련결한 실험이 시작되면서부터 인터네트연구의 력사가 시작되였다. ARPA는 파케트교환이라고 부르는 새로운 기술에 의하여 여러 장소들에 있는 콤퓨터들이 통신할수 있겠는가 하는데 관심을 가졌다.

ARPA는 Advanced Research Projects Agency의 략자이다. 이 기술은 여러대의 콤퓨터들이 서로 선으로 련결될 때 한개의 통신선로를 공유하게 한다.

파케트교환망은 쪼각 또는 파케트들로 분리된 정보를 보낸다. 파케트들은 적당한 목적지에 송신되며 그림 24-1에서 보여 준것처럼 그것들을 사용하는 수신콤퓨터에서 적당하게 재정돈된다.

매 파케트들은 그것들의 목적지에 대한 정보를 포함하며 정보는 파케트들로 분할되여 망을 넘나든다. 많은 콤퓨터들은 자료통로에서 운반수단과 같이 파케트들을 적재하여가지고 한개의 선을 공유한다. 이것은 파케트교환망을 실현할수 있게 한다.

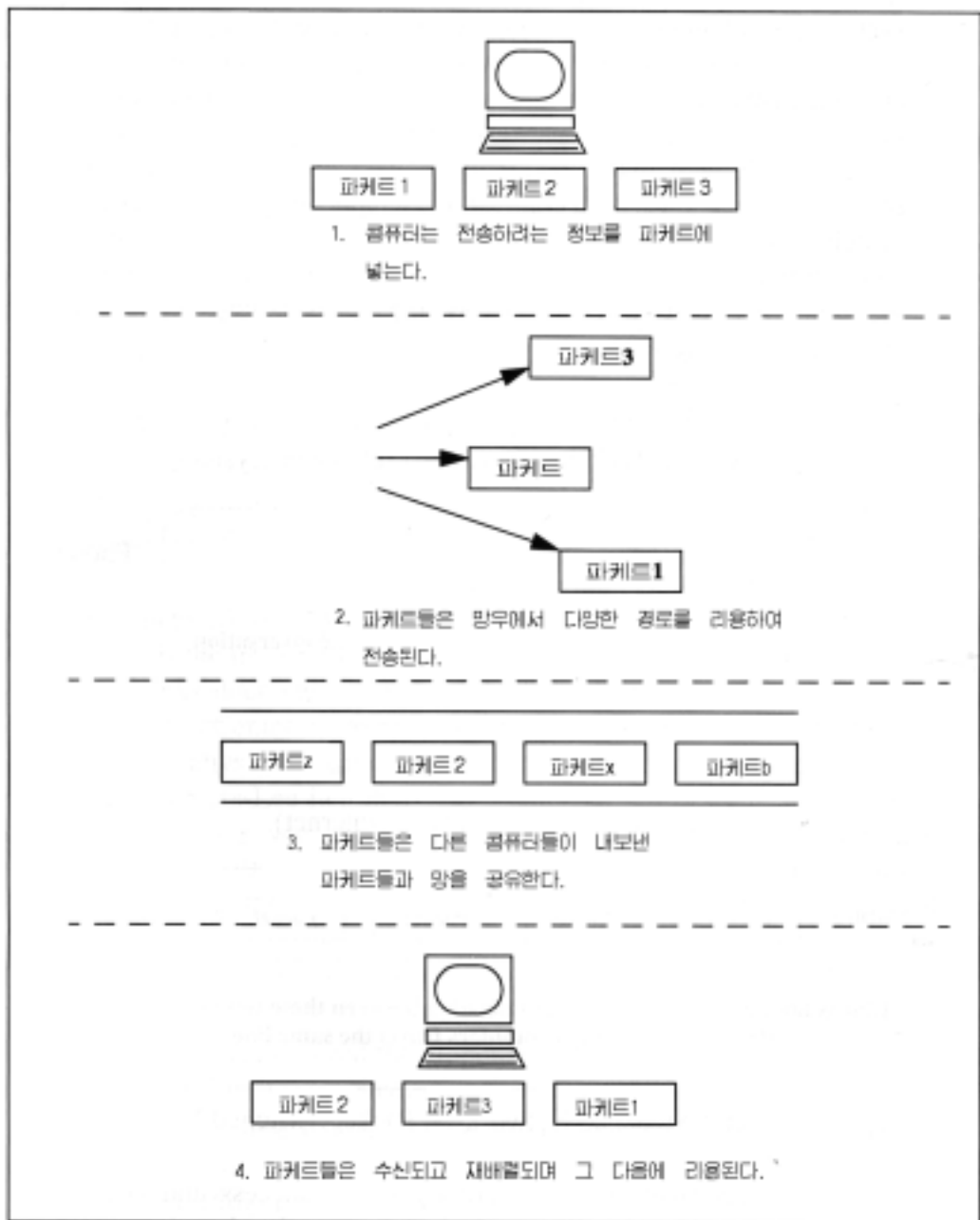



그림 24-1. 파케트교환망

전화체계를 구성하는 회선교환망과는 달리 파케트교환망은 한개의 선을 요구하지 않고 파케트수신자와 파케트송신자사이에 련결을 끊지 않도록 한다. 매 파케트들은 목적지로 향하는 많은 통로를 가질수 있다. 실례로 만일 어느 한 정보가 5 개 파케트로 구성되였을 때 매 파케트는 망에서 여러개의 통로를 가질수 있고 5 개 파케트들은 목적지에서

조립되며 사용된다. 비교해서 말한다면 전화체계는 망의 한부분으로서 한개의 접속을 가지는 회선교환망이다. 이 접속에 의하여 어떤 사람이 말하고 있는가 또는 말하지 않고 있는가가 알려 지게 된다. 그림 24-2에서는 회선교환망과 파케트교환망을 보여 준다.

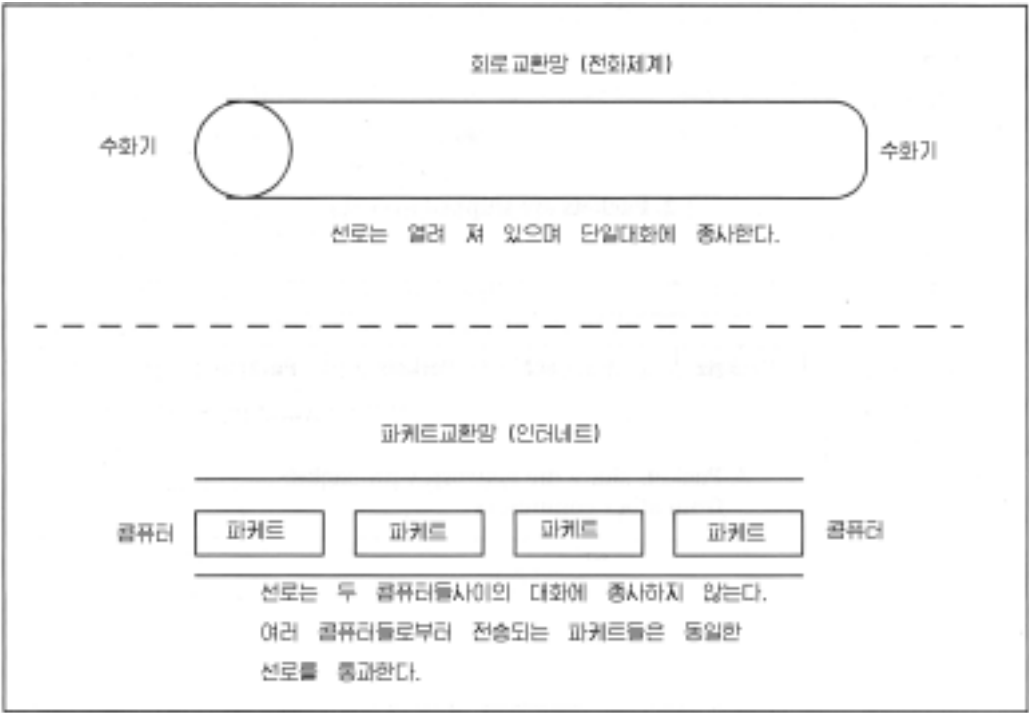



그림 24-2. 회선교환망과 파케트교환망

파케트교환기술은 성공한후에 ARPANet로서 알려 지게 되였다. 몇명의 진취성이 강한 대학생들에 의하여 **직결회의**(Online Conference)를 처리하는 방법이 개발되였고 그로 인해서 결국 사람들은 **직결통신**(Online Communication)의 능력을 인정하게 되였다.

1970년대에 ARPA는 여러가지 콤퓨터망들사이에 자료전송을 위한 규약들이나 리용규칙들을 개발하였다. 이러한 망들 호상간의 작업과 망들사이의 규약들은 인터네트를 개발할수 있게 하였다. 1970년대 말에는 여러 나라들에서 ARPA망과 이것과 류사한 망들사이에 련결을 진행하였으며 《거미줄》이 늘여 지는것과 같이 광범히 보급되였다.

인터네트는 1980년대에 매우 빠른 속도로 보급되였으며 대학, 정부기관, 연구기관들이 이 세계적인 망에 접속하기 시작하였다.

1990년대에 인터네트열람기출현은 망의 폭발적인 증대를 가져 왔다. 또한 자료를 창조적형태로 현시할수 있었고 인터네트는 전문가들과 애호가들의 관심을 끌기 시작하였다. 상업적인 인터네트봉사는 장치체계와 대등하게 빨리 발전하였다. 1994년 중엽에 이르러서는 매일매일의 사무처리를 인터네트로 하기 시작하였다.

인터네트는 열람기상에서 표시된 정보보다 더 많은 정보를 가지고 있다. 왜냐하면

인터네트가 다종다양한 자료를 전송하는 망이기때문이다. WWW(World Wide Web) 또는 Web 는 인터네트가 가장 빨리 발전한 부분이다. 이것은 문서들을 국부적으로, 원격적으로 련결하며 핵물리연구정보를 공유하기 위하여 스위스 제네바의 CERN(유럽핵연구쎈터)에서 개발된 인터네트장치체계이다. 그것을 제기한것은 1989 년 Tim Berners-Lee 이다. 그 후 1991 년에 첫 지령행열람기가 소개되였고 1993 년에 Voila X Windows 열람기가 나왔다. 여기서 Web 를 위한 첫 도형기능이 제시되였다. 같은 해에 CERN 은 자기의 마킨토시열람기를 소개하였고 National Center for Supercomputing Application(NCSA)은 Mosaic 의 X Windows 를 소개하였다. Microsoft 에 뒤따라 Netscape 도 시장에 자기의 열람기를 내놓았다.

열람기들은 인터네트의 대중적인 흥미를 자아내는 자료의 도형적표현과 형식을 처음으로 실현하였다.

오늘 대부분 새로운 사용자들은 Web 열람기를 통하여 인터네트와 호상작용한다. 그러나 이보다 앞서 지령행 UNIX 봉사프로그람들이 있었으며 아직도 사용되고 있다. 프로그람과 파일들은 Archie 봉사프로그람을 사용하여 목록화되고 정보는 FTP(File Transfer Protocol)를 통하여 받을수 있다. Telnet 는 프로그람을 실행하기 위한 인터네트상의 콤퓨터에 가입하기 위해 사용된다. Gopher 는 인터네트파일들을 서술하는 계층적인 안내로 사용되였고 Veronica 는 Gopher 싸이트에서 더 묘한 탐색을 진행하였다. 1994 년에는 Web 열람기의 초시기에 대략 500 개의 Web 싸이트들이 있었다. 오늘 매일과 같이 직결방식에 가입하는것으로 하여 수백만의 Web 싸이트가 있다.

인터네트의 의뢰-봉사모형

프로그람작성에 대한 설계는 여러 해에 걸쳐 발전되였다. 의뢰기는 사용자를 위한 대면부장치이며 봉사기로부터 정보를 받는다. 봉사기는 통보를 보내는 장치를 설정하는 장치이며 의뢰기로부터 정보를 받는다. 이 의뢰기-봉사기설계는 그림 24-3 에서 보여 주고 있다.

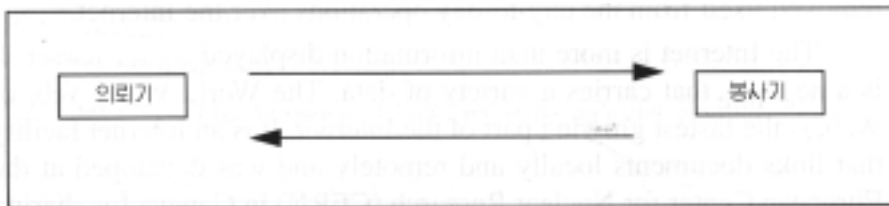



그림 24-3. 의뢰기-봉사기설계

초기설계에서 의뢰기로서는 저급한 말단들을, 봉사기로서는 고급한 콤퓨터를 쓰는것으로 되여 있었다. 저급한 말단들은 본문정보를 표시할수만 있으면 되였다. 후에 일부 도형적표현을 할수 있는 X-말단들이 소개되였다. 봉사기는 수처리와 알고리듬으로부터 마당확인에 이르기까지 거의 모든것을 처리한다. 이것은 마당이 확인되여야 할 때마다

상업적론리와는 반대로 표현되였다. 즉 내용들은 봉사기에로 망을 통하여 전송되며 확인하고 다시 되돌아 왔다. 마당에 오유가 있으면 이 공정은 반복되였다. 이 구조는 망통과량을 증가시켰고 사용자들은 응답에서 종종 지연당하군 하였다.

기술의 발전과 개인용콤퓨터들의 광범한 보급으로 인하여 개발단체들은 마당확인과 같은 론리가 의뢰기우에 존재하도록 실현하였고 미리 확인된 정보가 봉사기에 전송될수 있도록 실현하였으며 망통과량을 최소화하고 사용자의 응답시간을 증대시켰다. 의뢰기들이 더욱 세련되였기때문에 다양하고 화려한 도형대면부가 완성될수 있었다. 그러나 그것들은 값이 비쌌으며 봉사기프로그람에 따라서 복합의뢰기프로그람도 발전되여야 하였다. 의뢰기에 존재하는 모든 프로그람은 매 사용자들에게 배포되여야 하고 설치되여야 하였다. 이것은 의뢰기프로그람에 대한 그 어떤 수정도 사용자들과 토론되여야 하며 사용자들은 갱신된것을 알아야 한다는것을 의미하였다. 결국 인터네트는 사용자가 의뢰기프로그람의 새로운 방안을 쓸수 있도록 프로그람을 분배하기 위하여 리용될수 있게 되였다. 그러나 아직 사용자집단이 의뢰기프로그람에 대한 정확한 방안을 가졌는가를 담보하는데는 여전히 문제가 있다. 매 사용자가 자기의 프로그람을 갱신해야 하기때문에 이것은 사용자를 더 지원해 주는 개별적체계에서 오유의 가능성을 높여 주었다. 매 사용자가 새로운 프로그람작성을 실행하는데 많은 시간을 소비하기때문에 하나의 의뢰기프로그람을 새로 작성하는것은 가격이 매우 비싸다. 가령 매 사용자가 갱신에 10min ~ 20min 을 소비한다고 하면 거기에 수백 수천의 사용자수들을 곱하였을 때 한개 의뢰기를 갱신하는데 드는 시간은 굉장하다.

WWW 와 열람기들이 대중적인것으로 되였을 때 개발단체들은 자기들의 난점의 해결가능성을 보기 시작하였다. 열람기는 인터네트를 통한 표준지령모임을 받으며 이 지령들에 대해 정보를 표시할것을 지시한다. 그러므로 개발자들은 의뢰기프로그람을 더이상 개발하지 않는다. 그리고 사용자들은 의뢰기프로그람을 더이상 갱신하지 않아도 된다. 그들은 그대신에 열람기를 사용한다. 그러나 이것은 제한이 있다. 개발자들은 열람기가 제공해 주는 방법을 리용하여 오직 대면부만을 만들수 있다.

이 해답은 어느 하나도 완전무결하지 못했다. 열람기의 기능은 개발자들이 현재 익숙된 대면부보다 낮은 수준의 기능이였다. 처음에 열람기들은 오직 정적페지만을 제공하였다. 많은 의뢰기-봉사기응용프로그람들은 단추나 마당확인과 같은 자기 의뢰기로부터 동적특성을 필요로 하였다. 결국은 Java 와 Javascript 그리고 다른 새로운 기술들이 창조되였는데 이것으로서 동적 Web 페지들을 쓸수 있게 되였다. 현재 개발자들은 의뢰기프로그람으로써 열람기사용을 시작할수 있고 여러가지 기술을 결합함으로써 의뢰기프로그람에 대한 힘겨운 갱신이 없이도 Web 에서 충분히 기능적인 응용프로그람을 창조할수 있다. 이 기술은 아직 부족하며 제한을 가지고 있다. 개발자들은 처음에 단순한 의뢰기-봉사기응용프로그람들을 개발하였으나 기술이 발전함에 따라 더 복잡한 응용프로그람을 실현할수 있게 되였다.

인터네트를 사용할 때 의뢰기와 봉사기의 접속은 정보를 교환할 때에만 이루어 진다. 정보가 봉사기로부터 의뢰기로 전송된후에 이것들사이의 접속은 없어 지지만 인터네트와의 접속은 남아 있다.

규 약

규약은 자료의 송수신을 보장하는 장치와 프로그람에 대한 규칙들의 모임 또는 표준 규칙이다. 이것은 "대화"를 통하여 자료교환을 실현한다. 다음의것은 통신규약에 의해 처리되는 교환에 대한 개념을 준다.

<u>콤퓨터 1에서 콤퓨터 2에로 자료를 보내려고 한다.</u>

콤퓨터 1: 거기에 당신이 있습니까?
콤퓨터 2: 있습니다.
콤퓨터 1: 당신은 자료를 받을 준비가 되였습니까?
콤퓨터 2: 그렇습니다.
콤퓨터 1: 통보는 …입니다. (자료를 전송한다.) 당신은 그 통보를 받았습니까?
콤퓨터 2: 받았습니다.

콤퓨터 1: 자료를 더 보내겠습니다. (자료를 전송한다.) 당신은 그 자료를 받았습니까?
콤퓨터 2: 받지 못하였습니다.

콤퓨터 1: 다시 …을 보내겠습니다. (자료를 다시 전송한다.) 당신은 그 자료를 받았습니까?
콤퓨터 2: 받았습니다.

콤퓨터 1: 더이상 자료가 없습니다. 안녕히 계십시오.
콤퓨터 2: 안녕히 계십시오.

표면상으로는 단순하지만 이 생각은 세계적인 콤퓨터들과 망들이 인터네트에 정보와 통보를 공유하게 한다.

인터네트를 사용할 때 의뢰기와 봉사기사이의 접속은 정보교환이 진행되는 동안 유지된다. 정보가 의뢰기-봉사기사이에 전송된후 접속이 없어 진다. 그러나 인터네트련결은 남아 있다.

앞에서 말한바와 같이 자료는 파케트들로 분리되고 전송되며 목적지에서 결합된다. 이 과정은 인터네트상의 가장 중요한 두 통신규약인 TCP(Ttransmission Control Protocol)와 IP(Internet Protocol)의 일감이다.

TCP/IP

TCP/IP는 모든 형태의 콤퓨터들이 서로 자료를 전송할수 있도록 설계된 통신규약이다. 인터네트는 파케트교환망이기때문에 TCP/IP를 사용한다.

TCP는 접속지향규약이며 두 콤퓨터들사이에 접속을 실현하다. 그것은 정확한 자료전송을 보장하며 필요하다면 오유를 검사하고 자료의 재전송을 요구한다.

TCP/IP 는 **경로선택규약**(Routable Protocol)인데 이것은 모든 통보가 목적지주소를 포함하고 있다는것을 의미한다. IP 는 경로결정수법을 제공한다. 경로결정수법은 IP 주소를 사용한다.

IP 주소는 15.199.45.11 과 같이 점으로 분리된 4 개수로 이루어 져있다. 첫째 부분은 망주소 또는 망식별자이며 두번째 부분은 호스트주소 또는 호스트식별자이다. 매 의뢰기와 봉사기는 그것을 인터네트상에서 찾을수 있도록 IP 주소를 가져야 한다.

본질적으로 TCP 는 파케트들을 분리하고 결합하는 역할을 하고 IP 는 파케트들이 정확한 목적지에로 보내지도록 하는 역할을 한다.

HTTP

HTTP(Hyper Text Transfer Protocol: 하이퍼본문전송규약)는 WWW 에서 봉사기에 접속하도록 하는 통신규약이다. 그것은 Web 봉사기와 접속을 실현하여 주고 의뢰기열람기에로 HTML 페지를 빨리 전송하게 한다.

Web 주소는 http:// 의 HTTP 규약으로 시작한다. Web 열람기들은 규약이 기입되지 않으면 HTTP 규약에 기정적인 주소를 리용하게 한다. 실례로 Netscape 싸이트의 주소는 다음과 같다.

http://netscape.com

Netscape.com 이 HTTP 규약이 없이 기입되면 열람기는 그것을 기정적으로 지적하며 주소의 앞에 http://접두사를 붙인다.

봉사기와 의뢰기사이의 HTTP 접속은 의뢰기열람기에 자료가 전송되는 기간만 유지된다. HTTP 접속이 끝났어도 인터네트와의 TCP/IP 접속은 남아 있다. 그러므로 의뢰기는 여전히 인터네트에 접속되여 있지만 봉사기와의 접속은 더이상 없다.

HTTPS

HTTPS(HyperText Transfer Protocol Secure: 하이퍼본문전송규약보안)는 안전한 Web 봉사기에 접근하기 위한 규약이다. 대부분의 Web 봉사기들은 80 의 표준포구번호를 가진다. https://를 사용하는 Web 주소들은 표준포구번호보다 안전한 포구번호에 통보를 전송한다. 이 과정은 **보안규약**(Security Protocol)에 의해서 조작된다.

이 규약은 보안 Web 페지에 기입될 때 사용된다. https://앞붙이는 http://대신 Web 주소에 쓰인다.

Web 열람기

앞에서 언급한바와 같이 Web 는 의뢰기/봉사기모형을 사용한다. Web 열람기는 인터네트상에서 WWW 에 대면부로써 봉사되는 의뢰기프로그람이다. Web 에 기입된 정보나 자료들은 Web 열람기를 리용하여 찾을수 있으며 볼수 있다.

문서들은 도형, 본문, Java 애플레트와 같은 실행할수 있는 프로그람과 다른 문서에로의 련결 등을 포함하고 있다. 열람기는 자료를 읽어 들이며 사람이 리해할수 있는 형식으로 제시된다.

열람기들은 많은 여러가지 콤퓨터체계에서 쓸수 있다. 열람기가 선택될 때 그것은 콤퓨터체계와 호환성이 있어야 한다. 많은 콤퓨터체계들은 미리 설치된 열람기들을 가지고 있다. 대표적으로 Microsoft Windows 를 사용하는 개인용콤퓨터들은 Internet Explorer 열람기를 가지고 있고 UNIX 체계들은 Netscape 열람기를 가지고 있다. 일부 인터네트봉사업자들은 자기들의 독자적인 열람기프로그람을 제공한다.

Web 에서 대부분의 정보는 인터네트에 접속하고 있는 서로 다른 콤퓨터들에 놓여있는 련결페지들을 통하여 찾을수 있다. 봉사기로 불리우는 이러한 망콤퓨터들은 정보를 저축하고 배달한다.

정보는 Uniform Resource Locator 혹은 URL 을 통해서 요청된다. 망은 자료를 가진 봉사기를 찾고 복사를 요구하기 위하여 URL 을 사용한다. URL 은 규약의 앞붙이와 령역이름을 포함한다. 그것은 또한 포구번호, 보조등록부이름, 문서이름을 포함한다. 문서이름이 없으면 index.html 로 암시된다.

아래에서는 URL 의 성분들을 실례를 들어 설명한다.

URL: http//www.lizards.com:80/neon/products.html

프로토콜앞붙이: http://
령역이름: www.lizards.com
코드번호: 80/
보조등록부: neon/
문서이름: products.html

열람기들은 Web 상에서 문서를 요구하는 수단으로써 URL 들을 접수한다. URL 들은 열람기에 기입할수도 있으며 련결이라고 알려 진 문서의 부분으로서 현시될수도 있다. 사용자는 련결령역에 마우스를 누름으로써 그 문서에로 뛰여 넘어 갈수 있다.

URL 이 기입되였거나 련결이 눌러 졌을 때 령역이름은 IP 주소로 표현된다. IP 주소는 URL 에서 령역이름대신에 사용될수 있지만 수자들의 렬보다 이름을 기억하는것이 쉽기때문에 통속적이지 못하다. IP 주소는 알맞는 봉사기를 찾는데 리용되며 그것이 문서의 복사를 요구하게 한다. 봉사기는 Web 를 통해 복사물을 보내여 사용자콤퓨터에 전달한다. 자료가 도착했을 때 열람기는 그림을 현시하고 Java 애플레트들을 실행하면서 자료를 문서로서 해석한다.

제２５장. Java

프로그람작성언어인 Java(쟈바)는 1995 년 이후 소개된 때로부터 대단한 인기를 끌었다. Java 는 초기에 각종 소비전자장치들을 위한 응용프로그람을 만들기 위하여 생겨 났다. WWW 의 급속한 발전으로 개발자들은 정적 Web 페지보다 더 좋은것을 요구하였다. Java 의 개발자들은 Java 와 같은 독립적인 구조를 가진 언어를 가지고 이러한 기술적난점을 극복하였다. 인터네트가 여러가지 기계에서 실행하고 있기때문에 그러한 모든 기계들에서 프로그람을 수행시키는 능력을 가지는것이 아주 중요하였다. 이것이 Java 가 세계적으로 널리 보급될수 있는 기회로 되였다.

Java 의 개발자들은 대부분의 프로그람작성자들이 배우기 쉽고 친숙한 프로그람언어를 만들려고 하였다. C 와 C++언어들은 이미 널리 보급되고 사용되였다. Java 를 설계할 때 대부분의 C 와 C++구조들을 리용하였는데 이때 C 와 C++의 많은 특징들이 없어 졌다. 없어 진 대부분 특징들은 프로그람작성실천이 약하거나 드물게 사용되는것들이였다. 한가지 실례는 Java 가 지적자를 사용하지 않는다는것이다. 많은 프로그람작성오유들은 빈약한 지적자실현에 있으므로 Java 에서는 그것을 제거하게 되였다.

구성방식독립성

C 와 C++로부터 발생되는 실행파일들은 그것들이 콤파일된 콤퓨터가동환경에서만 실행된다. 이 실행파일들은 서로 호환성이 없다. 서로 다른 기종들이 광범히 집결되여 있는 인터네트세계에서 이 사실은 문제로 된다. 많은 경우에 C 와 C++의 원천코드조차도 주어 진 콤퓨터가동환경에서 콤파일되지 않는다.

Java 는 구성방식의 독립성을 가진다. 그것은 콤파일하기도 하고 해석하기도 하는 언어이다. Java 콤파일러는 본래의 기계코드보다 오히려 중간구조를 가진 바이트코드를 발생한다. 바이트코드는 그것이 콤파일된 가동환경과는 무관계하다. 이것은 다중가동환경에 Java 프로그람을 쉽게 전송할수 있게 한다.

Java 프로그람들은 Java 가상장치를 가진 콤퓨터의 가동환경에서만 실행될수 있다. 이것은 Java 프로그람들이 원천코드로 재콤파일이나 변환이 없이 실행할수 있게 해 준다. Java 바이트코드들이 콤퓨터체계에 온 다음 JVM 에 있는 Java 해석기는 그것들을 본래의 기계어코드로 해석한다.

그림 25-1 에서는 콤파일되고 해석되는 Java 의 특징에 대하여 설명한다. Java 의 원천코드는 콤퓨터 A 에 있으며 바이트코드로 콤파일된다. 인터네트를 경유하여 바이트코드가 요구될 때 복사가 망을 통해 전송된다. 콤퓨터 B 는 요구를 가지고 바이트코드의 복사를 접수한다. 바이트코드는 Java 프로그람을 실행하여 콤퓨터 B 가 리해할수 있는 본래의 기계코드로 콤파일된다.

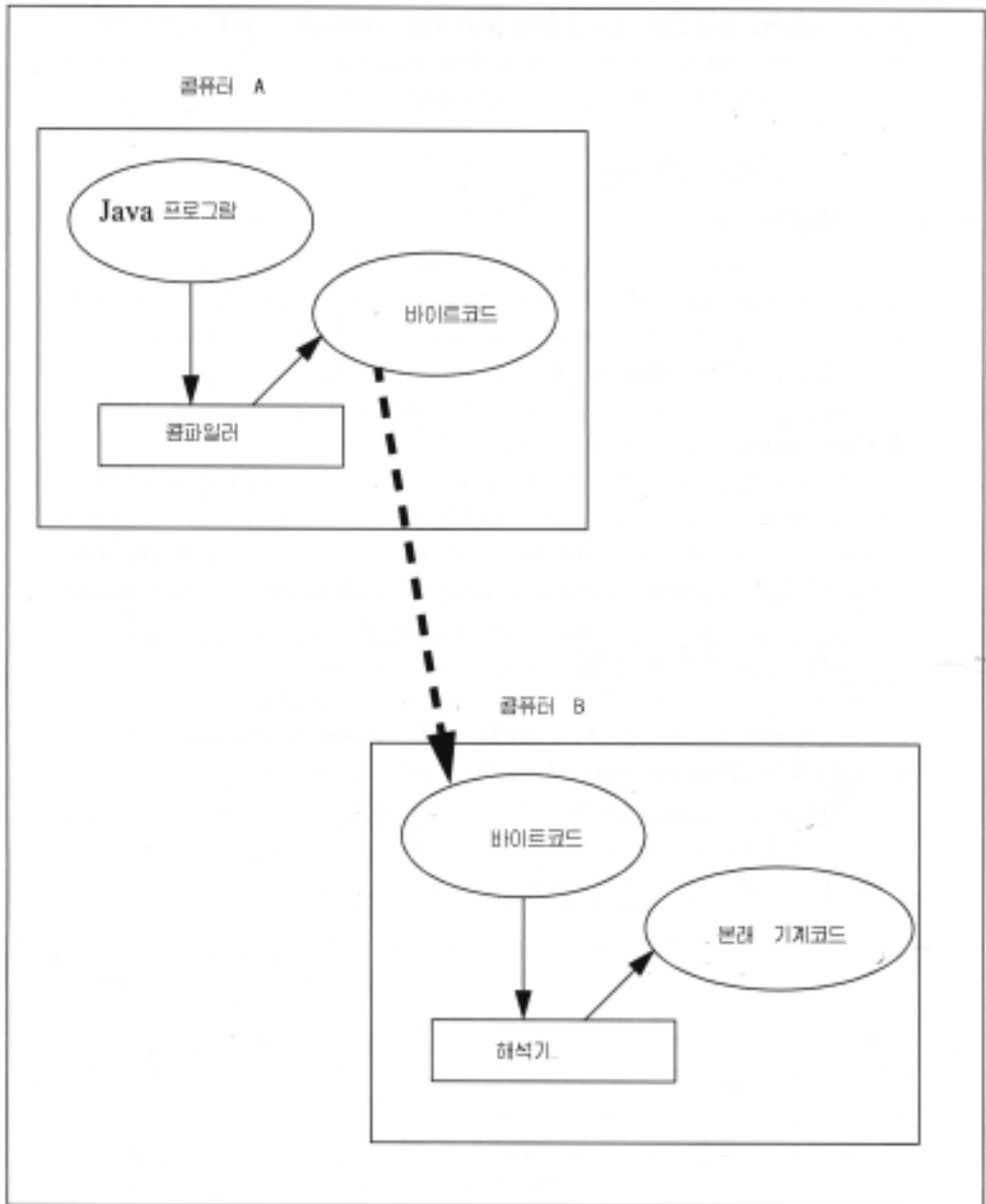



그림 25-1. Java가 콤파일되고 해석되는 과정

모든 Java 프로그람들이 망을 경유하도록 작성되지는 않는다. 많은 프로그람작성자들은 호환성이 있는 프로그람을 만들기 위해 Java 를 리용한다. 이것은 재번역을 방지하는데도 쓸모가 있으며 일부 경우들에 다른 언어들이 서로 다른 콤퓨터가동환경상에서 실행할수 있도록 고쳐 쓰기를 하는것을 방지하는데도 쓸모가 있다.

Java 가동환경

가동환경은 전형적으로 프로그람이 실행되는 장치와 프로그람환경이다. 이와는 반대로 Java 가동환경은 다른 장치가동환경들의 맨 우에서 실행되는 유일한 프로그람가동환경이다.

Java 는 두가지 요소 JVM 과 **Java 응용프로그람대면부**(Java API)로 이루어 져 있다. JVM 은 호환성을 가지지 않는다. JVM 들은 특별한 콤퓨터가동환경을 위하여 리용된다. JVM 들은 Java 바이트코드를 실행하기 위한 기계의존코드로 해석한다. Java 프로그람은 그것이 JVM 을 가지고 있지 못하면 콤퓨터체계상에서 실행할수 없다.

Java API 는 자주 사용되는 프로그람요소들의 거대한 집합이다. Java API 들은 련관된 요소들의 서고와 묶음들로 그룹화되여 있다. Java API 들은 개발자가 사용하기 위하여 미리 만들어 놓은 성분들이 있기때문에 개발을 빨리 할수 있게 한다.

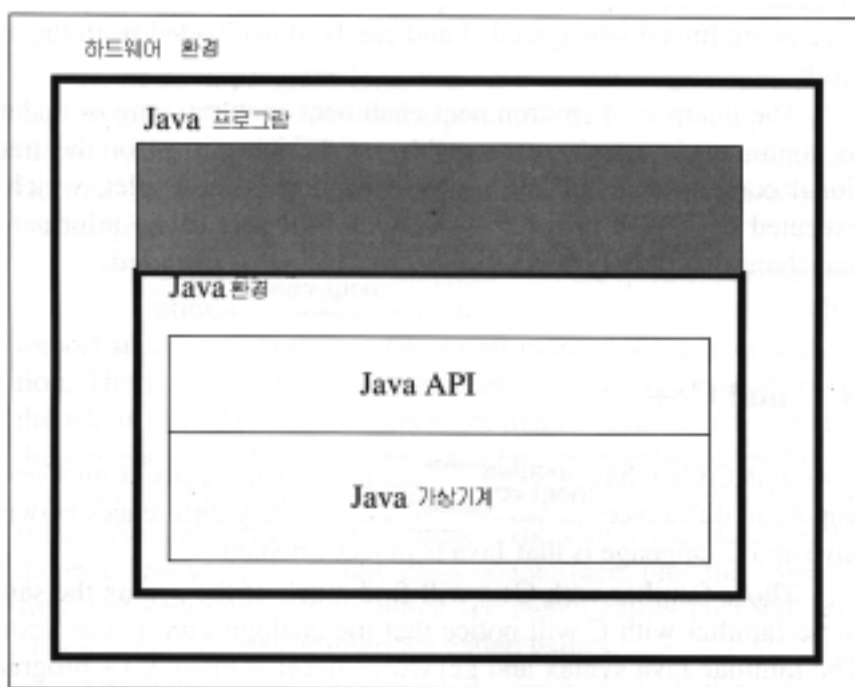



그림 25-2. Java 가동환경에서 Java 프로그람

그림 25-2 에서는 Java 가동환경에서 Java 프로그람이 실행되는 과정을 보여 주며 Java API 들과 JVM 이 프로그람을 콤퓨터가동환경의 의존성으로부터 어떻게 분리시키겠는가 하는것을 보여 준다.

Java 는 가동환경과 독립환경이지만 그것으로 인한 약점도 있다. Java 호환성이 바이트코드를 기계코드로 해석해야 하는 JVM 을 통하여 가능하기때문에 본래의 코드보다 속도가 더 느리다.

정확한 콤파일러들, 잘 조화된 해석기들, 더 개선된 콤파일러들, HotSpot 와 같은 기술들은 호환성에 대한 우려가 없이 Java 의 실행을 본래 코드의 실행과 더 가깝게 한다.

동적프로쎄스

C 또는 C++콤파일과정에 볼수 있었던 련결단계는 Java 환경에서 기본적으로 쓰이지 않는다. Java 의 련결은 실제로 클라스적재자에 의하여 새로운 클라스를 적재하는 처리이다. 이것은 더 확장되고 쉬운 처리이다.

Java 언어는 호환성이 있고 해석방식으로 처리하므로 동적처리수준이 높은 체계와 확장된 동적체계를 처리하게 한다. Java 언어는 환경에 적응하도록 설계되였다. Java 클라스는 필요할 때 련결되고 망으로부터 적재될수 있다.

해석된 환경은 전통적인 콤파일러와 련결주기를 기다리지 않고 계속 실행할수 있도록 하기 위하여 **원형**(Prototype)만들기와 갱신을 빨리 하게 한다. 례외로 되는것은 애플레트이다. 그것은 Web 열람기에서 수행된다. 정보와 변화들을 보관하는 대부분의 Web 열람기들은 열람기가 재기동되면 볼수 있다.

Java 와 C/C++

Java 와 C/C++는 많은 공통점을 가지고 있다. 그러나 중요한 차이점도 있다. Java 와 C 언어사이의 근본적인 차이점은 Java 가 객체지향언어라는데 있다. C++에서 친숙해 진 문법들과 같은것들이 많지만 C 에서 자주 쓰이는 문법들과는 그리 류사하지 못하다. 잘 알려 진 Java 문법과 실마리어들은 C++프로그람작성자들이 만든것들이다. 차이점은 언어에 대한 엄밀한 검사를 충분히 담보해 준다는것이다.

Java 환경

Java 에서 개발을 시작하기 위해서는 JDK(Java Developers Kit)에서 제공되는것과 같은 Java 개발도구들이 필요하다. JDK 는 장치제작자싸이트들로부터 제공 받는다. JDK 는 다른 요소와 함께 Java 콤파일러, Java API 들, JVM 그리고 애플레트보기 등을 제공한다.

Web 열람기들에서 언어로서의 Java 에 대한 대부분의 견해는 Web 에서 실행하는 프로그람을 개발가능하게 한다는것이다. 그러한 견해가 옳다고 할지라도 Java 도 역시 C 와 C++처럼 main( )방법을 가지고 봉사기에 쓸수 있는 응용프로그람언어이다.

Java 애플레트가 Web 열람기내에서 실행될 때 애플레트클라스가 main( )방법대신에 사용된다. Java 애플레트가 응용프로그람이 아니라 Web 열람기나 애플레트보기와 같이 이미 실행하고 있는 Java 응용프로그람에 의해 적재되고 실행되는 Java 클라스라는것을 주목해야 한다.

main( )방법은 Java 프로그람이 응용프로그람처럼 봉사기상에서 실행될 때 사용된다. 이 장에서는 애플레트클라스에 대하여 자세히 론의된다. 그 목적은 인터네트프로그람작

성법을 론의하기 위해서이다. 봉사기의 응용프로그람들에 대한 고찰은 몇개 단락에서만 취급한다.

Java 원천코드는 .java 확장자를 가진 파일로 존재한다. 그것은 선택적인 묶음명령문을 포함하고 있는데 그뒤에 임의의 개수의 입구명령문들, 한개이상의 클라스나 대면부정의들이 뒤따른다. 묶음명령문은 파일에 있는 코드가 어느 묶음의 부분인가 하는것을 규정한다. 입구명령문은 Java 클라스들을 현재 클라스에 리용할수 있게 한다. 한개이상의 클라스 또는 대면부가 정의된다면 클라스들중에 하나만이 public 로 선언 (묶음밖에서 사용할수 있도록) 된다. 원천파일이름은 public 클라스 혹은 대면부의 이름에 .java 확장자가 붙은것과 같다. 실례로 games. java 라고 불리우는 Java 원천파일은 오직 games 로 불리우는 한개 public 클라스 혹은 대면부를 포함할것이다.

Java 콤파일러는 < file >.class 에 바이트코드로 이루어 지는 객체파일들을 발생시킨다. Java 원천파일에서 매 클라스나 대면부정의는 .class 확장자를 가진 개별적인 파일들로 콤파일되며 그것을 정의하는 대면부와 같은 이름을 가진다. 실례로 Solitaire 클라스는 Solitaire.class 파일로 기억된다.

지령행인수 javac 는 Java 원천파일과 함께 콤파일될 때 사용된다.

javac [options] files

정확한 인수에 대해서는 사용자의 JDK 제작자와 론의하기 바란다.

봉사기에서 해석기를 호출하여 Java 응용프로그람을 실행하기 위하여 클라스이름과 임의의 인수들이 첨부되는 java 지령행인수를 사용한다.

java [interpreter options] classname [program arguments]

해석기의 오유수정방안을 실행하기 위하여 java_g 를 사용한다.

java_g [interpreter options] classname [program arguments]

main( )방법은 Java 프로그람을 시작하기 위하여 Java 해석기에서 사용된다. 그것은 다음의 소개로 시작된다.

public static void main(String args[ ])

Java 해석기는 main( ) 방법이 끝나거나 귀환될 때까지 실행을 계속한다.

main( )방법에 대한 인수는 args 또는 argv 라고 부르는 기호렬배렬이다. C 에서 argc 처럼 통과되는 배렬의 길이는 argv.length 로서 이것은 모든 Java 배렬들에서 쓸수 있다. 응용프로그람에서 지령행인수를 처리할 때 C 프로그람작성자들이 기대하는것처럼 인수배렬의 첫 요소가 클라스이름이 아니라는데 주의해야 한다.

main( )방법은 void 를 귀환하도록 소개되여야 한다. Java 프로그람으로부터의 값을 귀환하는것은 main()방법에서 귀환명령문으로 할수 없다. 귀환값이 필요하다면 옹근수값을 가지는 System.exit( )를 사용한다.

Java 는 조작체계환경변수를 읽을수 없다. 왜냐하면 그것들이 가동환경에 의존하기때문이다. 대신에 Java 는 체계속성목록을 가진다. 이것은 본문값들이 이름들과 관련되게

한다. 속성들은 응용프로그람동작수행의 손 쉬운 변경을 하도록 이 목록에 첨부될수 있다.

이름공간

Java 는 전체 인터네트범위에서 모듈의 동적적재를 지원하도록 설계되였기때문에 이름공간충돌을 피하는데 특별한 관심을 돌려야 한다. Java 는 아무런 대역변수들, 함수들, 틀들을 가지지 않는다.

클라스파일들이 기억된 등록부구조는 묶음이름과 같은 요소들을 가진다. 실례로 클라스의 어떤 완전한 이름이

gameworld.games.multi_player.doom.landscapes

라면 클라스파일의 완전경로는 다음과 같다.

gameworld/games/multi_player/doom/landscapes.class.

파일이름은 Java 클라스경로와 련관하여 해석된다.

Java 클라스경로는 Java 해석기가 사용자정의클라스를 찾을수 있도록 개발자에 의해 설정된 환경변수이다. 해석기는 표준체계클라스가 어디에 설치되였는가를 알고 있으며 환경변수속에서 지정된 경로의 마지막위치를 추가할수 있다. 사용자는 우의 실례에 대한 클라스경로를 다음과 같이 설정할수 있다.

```
setenv  CLASSPATH  .:/home/gameworld/games/multi_player/doom
```

설 명 문

Java 에는 세가지 설명문형식 즉 /*로 시작되고 */로 끝나는 표준 C 형식과 //으로 시작되여 행의 끝까지 계속되는 C++형식, 특별한 doc 설명문이 있다. doc 설명문은 /**로 시작되고 다음에 */까지 계속되며 닫기게 하지 말아야 한다. doc 설명문은 Java 원천코드로부터 단순한 온라인 문서를 처리하기 위하여 javadoc 프로그람으로 처리한다.

전처리기가 없다

Java 는 어떤 전처리기를 가지고 있지 않으며 #define, #include, #ifdef 를 지원하지 않는다.

상　　수

Java 에서 마지막으로 소개되는 변수는 상수이며 값은 절대로 변하지 않는다. C 의 #define 과 동등한 Java 상수는 클라스정의에서 마지막으로 소개되는 정적변수이다.

마크로가 없다

Java 에는 함수호출을 머리부에 보관해 두는 C 마크로와 같은것을 가지지 않는다. 많은 Java 콤파일러와 가상장치들은 적당한 곳에 짧막한 java 방법들을 묶어 놓고 있다. 이렇게 묶어 놓은 방법은 개발자가 방법들에 대한 조종에 무관계하도록 한다.

포함파일이 없다

Java 는 #include 지시자를 가지고 있다. 왜냐하면 클라스이름들이 충분히 제한되여 있고 Java 콤파일러가 특별한 지시자를 사용하지 않고도 그것들이 어디에 있는지 알고 있기때문이다.

자 료 형

byte 와 Boolean 의 기초적인 형들이 C 의 표준형모임에서 Java 에 의해 추가되였다.

Java 도 역시 형의 크기와 부호를 엄격히 정의한다. C 에서 float 는 16, 32, 64bit 들이고 char 는 가동환경에 따라 signed 또는 unsigned 으로 리용된다. Java 에서는 크기와 부호가 가동환경을 넘어서도 그대로 보존된다.

C 에서 초기화되지 않은 변수는 보통 그 값으로서 불필요한 값을 가진다. Java 는 모든 변수들에 값을 기정적으로 설정해 준다. 다음의 표 25-1 에서는 자료형과 기정적인 값과 크기를 보여 준다.

표 25-1　　자료형, 값, 크기

자료형	기정값	크기
boolean	false	1bit
char	\u0000	16 bit
byte	0	8 bit
short	0	16 bit
int	0	32 bit
long	0	64 bit
float	0.0	32 bit
double	0.0	64 bit

옹근수형

Java 옹근수형에는 byte, short, char, int, long 들이 있다. char 를 제외한 모든 옹근수형들은 부호를 가진다. 실마리어 unsigned, long 과 short 는 더이상 문법적 효력을 가지지 않는다. 상수 long 은 문자 *l* 또는 L 을 첨가한다.

참조자료형

Java 의 원래 문법에서 정의되지 않은 자료형은 객체와 배렬인데 "참조자료형"이라고 부른다. 이 형들은 참고로 처리되는데 이것은 객체 또는 배렬의 주소가 변수에 기억되고 방법에 통과된다는것을 의미한다. C 에서 참고에 대한 값은 연산자 &를 사용하여 처리할수 있으며 *와 ->들의 사용은 참고되지 않는다. 이 연산자들은 Java 에 없다.

변경

변경은 변수와(또는) 방법선언들에 보충적인 정보나 제한을 제공하기 위하여 리용된다. Java는 수정실마리어들의 개수를 정의한다.

최종적인 실마리어는 클라스, 방법, 변수들에 대하여 사용된다. 마지막클라스는 부분클라스가 아니며 마지막방법은 덧쓰기할수 없고 마지막변수는 그 값을 설정할수 없다.

본래 실마리어는 방법선언을 하기 위한 변경이며 C 혹은 다른 종속되여 실행되는 가동환경의 다른 곳에서 방법이 사용된다는것을 지적한다.

동시성실마리어는 클라스가 개별적으로 안전하지 못하다는것을 지적하고 클라스의 동시적변경을 막는다.

일시적인 실마리어는 클라스안에서 요구되는 마당들에 대하여 사용된다. 이것은 마당이 객체의 지속상태부분이 아니며 객체와 함께 련결될 필요가 없다는것을 지적한다.

변하기 쉬운 실마리어는 동기화된 줄거리가 한개 마당을 사용하며 콤파일러가 그것과 합리화되지 않는다는것을 지적한다. 실례로 그것은 변수값을 기억기로부터 매번 읽어들이며 탄창에 기억된 복사물을 가지지 않는다는것을 지적한다.

지적자가 없다

Java 는 객체의 참조 또는 비참조를 자동적으로 처리한다. Java 는 개발자가 지적자 또는 각종 기억주소들을 다루지 못하게 한다. Java 는 객체나 묶음참조들을 옹근수로 바꾸어 놓거나 그 반대의 과정을 할수 없게 한다. Java 는 지적자연산을 허용하지 않으며 sizeof 연산자를 가지지 않는다.

Null

모든 참조형의 변수들은 기정으로 Null 값을 가진다. 그 의미는 "객체가 없다." 또는 참조가 없다는것을 의미한다. C 에서 Null(빔)은 0 으로 정의된 상수이다. Java 는 Null 을 실마리어로 만든다.

구조체 혹은 공용체가 없다

struct 와 union 형은 제공되지 않는다. 다음과 같은 두가지 문제에 주목하는것이 중요하다. 첫째로 클라스는 본질적으로 더 많은 특징을 가진 구조체이며 둘째로 union 은 부분클라스를 사용하여 모의될수 있다.

렬거형이 없다

Java 는 C 의 실마리어 enum 를 제공하지 않는다. 렬거형은 정적인 마지막상수값을 사용하여 부분적으로 대신한다.

TypeDef 가 없다

열쇠단어 typedef 는 형이름을 정의하는데 Java 에서는 제공하지 않는다. 왜냐하면 Java 가 더 단순하게 형이름을 만드는 구조가 있기때문이다.

객체창조

새로운 객체를 창조하기 위하여 객체클라스 혹은 형과 선택항목목록을 가지는 new 실마리어가 사용된다. 그 인수들은 클라스에 대한 constructor 방법에 넘겨 지며 이것은 새로운 객체내부변수를 초기화한다.

다음 실례에서는 클라스창조방법을 보여 준다.

```
TempBuffer s=new TempBuffer (name);
```

기호렬들이 너무 자주 창조되기때문에 Java 는 기호렬객체를 창조하기 위하여 다음과 같은 지름길을 리용한다.

```
String s="346 Ocean Blvd";
```

새롭게 창조된 객체를 위하여 필요한 기억기는 동적으로 할당된다.

Java 에서 new 로써 객체를 창조하는것은 C 에서 malloc( )를 호출하는것과 동등하며 C++에서 new 연산자를 사용하는것과는 더 류사하다.

객체접근

사용자는 객체마당에로 접근하기 위하여 점을 사용한다. 다음의것은 점을 사용하여 객체의 마당에 값을 주는 실례이다.

```
Circle c=new Circle( );
c.radius=2.0;
```

페품수집

Java 는 new 를 사용하여 객체를 창조하고 공간을 배치하게 한다. 그러나 공간을 자유롭게 일치시키는 방법은 없다. Java 객체가 더이상 사용되지 않을 때 자동적으로 검사하기 위한 **페품수집**(Garbage Collection)이라는 기술을 사용한다. Java 는 객체가 그것에 대한 참조가 없을 때 더이상 사용되지 않는가를 결정한다.

배 렬

객체와 같이 배렬은 참조에 의해서 조종되고 new 로서 동적으로 창조되며 그것들이 더이상 련관이 없을 때 자동적으로 페품이 수집된다. 배렬을 창조하는데는 두가지 방법이 있다. 첫번째 방법은 아래와 같이 배렬의 크기를 지적하면서 New 를 사용하는 방법이다.

```
int lottery_numbers[ ]=new int [6];
```

배렬에서 이 방법으로 창조된 요소들은 기정적인 자료형으로 자동적으로 설정된다. 두번째 방법은 초기화된 값들로 배렬을 창조하는데 C 에서 하는것과 같다.

```
int lottery_numbers[ ]={1, 7, 11, 17, 26, 29};
```

이 방법은 지적된 값으로 배렬의 요소들을 초기화한다.

배렬에서 요소들을 호출하는것은 C 와 C++에서와 같다. 즉 배렬에서 이름뒤에 있는 각괄호안에는 옹근수값식을 쓴다.

기 호 렬

Java 는 C 에서처럼 NULL 로 끝나는 문자배렬이 아니라 기호렬클라스의 경우와 같다. 기호렬문자들은 널리 리용된다. 그러므로 Java 는 기호렬문자들을 C 에서 하는것처럼 인용부호사이에 넣는다.

기호렬클라스는 내용을 변화시키는 방법을 포함하지 않는다. 만일 내용이 수정될것을 요구하면 String Buffer객체를 사용한다.

for 순환

Java 에서 대부분의 조종명령들은 C 와 C++에서와 같다. 의미 있는 차이를 가지는 명령문은 for 순환이다. Java 의 for 순환과 C 의 for 순환사이에는 두가지 차이만이 있다. 첫째 차이는 Java 에 단순식을 련결하여 다중식을 만들게 하는 반점연산자가 없다는것이다. Java 는 순환문법의 초기화와 증가부분에서 여러개 반점으로 분리된 식을 허용하지만 검사부분에서는 허용하지 않는다.

두번째 차이는 순환의 초기화부분에서 국부순환변수를 소개하는 C++능력과 관계된다는것이다. 즉

```
for（int x=0; x<max_shells; x++）
System.out.println（“current number of shells=”+x）;
```

선언된 변수가 그것의 범위로서 for 순환을 가진다고 할지라도 Java 콤파일러는 이미 존재하고 있는 변수 혹은 파라메터와 같은 이름을 가진 순환변수의 소개를 허용하지 않는다.

례외와 례외처리

례외처리는 Java 의 새로운 성질이다. Java 에서의 례외처리는 C++와 류사하다. 례외는 오유와 같은 일부 례외적인 상태가 일어 났다는것을 지적하는 신호이다. 례외를 내보내는것은 례외상태를 알리는것이다. 례외가 포착될 때 그것으로부터 회복하기 위하여 필요한 작용에 따라서 처리해야 한다.

례외가 발생하는 코드블로크가 례외를 포착하지 못하면 그것은 다음의 더 높은 코드블로크에로 이동한다. 만일 례외가 포착되지 않으면 그것은 마침내 main 방법으로 이동하게 되며 Java 해석기가 오유통보와 탄창자리길을 표시하고 탈퇴하게 한다.

Java 에서 례외는 Error 와 Exception 표시하는 두개의 표준부분클라스들을 가지는 객체이다. 부분클라스들 Error 와 Error 로 이루어 지는 임의의 례외들은 일반적으로 오유처리를 하지만 포착되지 않는 오유도 있다. 례외부분클라스들 Exception 에서 지적하는 례외들은 배렬초과와 파일의 끝과 같은 포착되고 회복될수 있는 상태들을 지적한다.

try/catch/finally 명령문들은 Java 의 례외조종기구이다. Try 은 례외를 가지는 비정상적인 탈퇴조종코드블로크를 만든다. try 블로크는 0 혹은 그이상의 포착항목에 의하여 산생된다. catch 항목은 조건적으로 "cleam-up"라는 코드를 포함하는 마지막블로크의 뒤에 온다. 즉

Try/catch/finally 명령문은 다음과 같이 나타난다.

```
try {
    //임의의 문제외에 표준적으로 실행할 코드블로크
    //그러나 실행되는 코드이다.
}
catch（AnException e）{
    //례외객체를 만나면 조종한다.
}
finally {
    //이 코드는 항상 그것이 어떻게 되여 있는지 무관계하게 try
    //부분전에 실행한다.
    //이 부분은 표준적으로 실행하든가
    //례외를 만나든가, 례외를 만나지 않든가
    //또는 코드가 break, continue, return 명령문때문에 탈퇴되는가 하는
    //것을 규정한다.
}
```

여러가지 코드부분들은 례외가 일부 요소들에서 나타날수 있다는것을 예상할수 있다. 실례로 프로그람이 사용자에게 옹근수를 입력할것을 요구할 때 옹근수가 아닌 다른 형의 수가 입구될수 있다. 다음 코드부분은 옹근수가 지령행에서 인수들을 거쳐 main( )방법으로 통과될것을 요구한다.

```
try {i=Integer.parseInt（argv [0]）;}
catch（NumberFormatException e）{
    System.out.println（“Argument must be an integer.”）;
    return;
}
```

애플레트

애플레트는 Web 열람기나 애플레트보기에 의해서 실행되도록 설계된 자그마한 응용프로그람의 종류이다. 그것들은 두가지 측면에서 다른 응용프로그람과 차이난다. 첫째로, 애플레트를 조종하는 많은 보안제한이 있으며 둘째로, 그것들이 할수 있는것들이 있다. 애플레트는 코드를 확인할수 없다. 실례로 같은 국부파일체계에서 처리가 허용되지 않는다.

프로그람작성자들은 애플레트와 응용프로그람들사이에 또 다른 차이가 있다는것을 알아야 한다. 애플레트는 main( )방법을 가지지 않는다. 애플레트는 Web 열람기나 애플레트보기와 같이 이미 Java 를 실행하고 있는 프로그람부분으로 적재되고 있는 부분클라스이다.

애플레트가 어떻게 작업하는가? 첫째로, Java 클라스는 애플레트클라스를 사용하여 창조된다. HTML 파일이 창조되면 HTML 태그는 애플레트를 참조하기 위하여 사용되고 인터네트상의 장치에서 그것을 요구하기 위해 사용된다. 그것은 열람기 또는 애플레트보기로 적재되고 실행된다. 처리는 사용자가 애플레트를 만들어 보면 더 명백하다.

첫 애플레트창조

첫 단계는 Java 원천파일을 창조하는것이다. 다음과 같은 Java 코드로 Hello World.java 라는 이름을 가진 파일을 창조한다.

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint (Graphics g) {
            g.drawString (“Hello World!!”, 50, 25);
    }
}
```

다음 Java 콤파일러를 사용하여 원천파일을 콤파일한다. 콤파일이 성과적으로 되였으면 HelloWorld.class 라는 파일이 같은 등록부에 창조된다.

애플레트를 수행하기 위하여 HTML 파일은 창조될것을 요구한다. HelloWorld.class 와 같은 등록부 Hello World.html 이라는 이름을 가진 확장자를 창조한다. 다음 본문을

입력한다.

```
        <HTML>
        <BODY>
        <APPLET code="HelloWorld.class" WIDTH=150 HEIGHT=50>
        </APPLET>
    </BODY>
</HTML>
```

애플레트를 실행하기 위해 HTML 파일은 Java 애플레트를 실행할수 있는 프로그람으로 적재될것이 필요하다. 실례로 JDK 에서 제공된 Java-열람기와 애플레트보기이다. 파일을 적재하기 위해 HTML 의 URL 파일을 입력하여야 한다. 아래에서는 이에 대한 실례를 보여 준다.

```
file/home/kit/applet/HTML/HelloWorld.html
```

애플레트보기를 사용할 때 지령은 다음과 같이 나타난다.

```
appletviewer file:/home/kit/applet/HTML/HelloWorld.html
```

클라스와 묶음의 반입

애플레트의 첫 두 줄은 java.applet.Applet 와 java.awt.Graphics 을 입구한다. 이 행들이 포함되지 않으면 다음의 변화들이 애플레트원천코드에 첨부되여야 한다.

```
Public class HelloWorld extends java.applet.Applet {
    Public void paint (java.awt.Graphics g) {
            g.drawString ( "Hello World!!", 50, 25);
    }
}
```

클라스의 입구는 임의의 앞붙이가 없이 프로그람에 관계되게 한다.

애플레트부분클라스의 정의

애플레트의 원천코드에서 실마리어 extends 는 이 경우에 HelloWorld 가 그것에 뒤따르는 클라스의 부분클라스이라는것을 지적한다. 애플레트는 애플레트클라스로부터 기능을 이어 받는다. 즉 더 중요한것의 하나는 열람기의 요구들에 대응하는 능력이다.

애플레트는 한개 클라스를 정의하는데서 제한되지 않는다. 그것은 사용자에게 유리하게 변경된 클라스를 정의할수 있게 한다.

도형사용자대면부로서의 애플레트

애플레트의 특징은 사용자대면부(UI)요소들의 사용을 촉진시키는것이다. AWT 와 Swing 서고들은 미리 만들어 진 도형성분을 가지고 있다. AWT 서고들은 첫번째 도형성분의 일부이다. 다음의 Swing 서고는 요소들의 도형보기와 감각을 사용자에게 편리하게 해주는 능력을 제공해 준다. 실례로 서고 Swing 는 가동환경과 관계없이 도형요소들에 대해 Windows 보기와 감각을 하게 해준다.

기본요소들은 단추, 검사창, 본문마당의 단순행, 구역편집, 마당표식, 목록들, pop-up 목록들, **미끄럼대**(Slider)와 흐름띠와 차림표들이다. 설계관리자는 개발자가 성분들을 배렬하고 잘 설계된 애플레트들을 만들어 낼수 있게 하여 준다.

Java 는 많은 API 들과 미리 씌여 진 성분들을 가진 서고들을 가진다. 더 배우려는 사람들은 이 서고에 대하여 연구해야 한다. 이 부분과 관련된 내용은 많은 책들에서 서술되여 있다.

제26장. Perl에 대한 소개

실용적인 추출 및 보고서작성언어(Perl)

Perl (Practical Extraction and Reporting Language)은 프로그람작성언어들중에서 가장 많이 쓰이는것중의 하나이다. 왜냐하면 여러가지 리유로 UNIX와 Windows체계조직스크립트, 일반적인 CGI(Common Gateway Interface), 자료기지처리, 파일처리 등을 포함하기때문이다.

Perl은 sed와 swk와 같은 쉘프로그람작성의 가장 좋은 부분들을 가지고 있다. Perl스크립트들은 보통 쉘프로그람보다 더 빠르고 더 간편하다.

Perl은 모든 UNIX 변종들 Linux(이 장에서 실례를 실행한 가동환경), Macintosh, Windows를 포함하고 있는 가동환경종류에서 실행할수 있다. 때때로 **노트형콤퓨터**(Notebook Computer)에 기초한 Windows들에서 Perl을 리용하고 있으며 UNIX체계상에서도 그것들을 사용하고 있다. Perl은 이 점에서 대단히 간편하고 좋다고 인정하였다. 인터네트상에는 Perl과 관련된 풍부한 자원이 있다. Web싸이트에서 가장 쓸모 있고 좋은 두가지 측면은 www.tpj.com(Perl Journal)와 www.Perl.com(일반적인 Perl 정보와 Perl의 내리적재)이다.

Perl은 일부 Web싸이트에서 자유롭게 리용될수 있다. 사용자는 쓸모 있는 Perl의 확장방안들을 리용하려고 할것이다. 그러나 Perl 그자체는 자유롭게 리용되며 거의 임의의 플레트홈에 쉽게 내리적재된다.

Perl은 쉘프로그람처럼 Perl 프로그람들이 콤파일되지 않았지만 해석된 언어이다. Perl 프로그람은 문법오유를 가지는 프로그람이 실행기간에 실패되는 쉘프로그람과는 달리 그것들이 실행되기전에 해석기에 의하여 완전히 해석된다.

이 장에서는 Perl의 기능에서의 일부만을 훑어 본다. 전체 Perl부분은 많은 Web싸이트, 참고서, 교육과 다른 재료들을 통하여 볼수 있다. Perl에 대한 많은 기능을 다는 소개할수 없으므로 이 장에서는 단순한 Perl을 시작하기 위한 기초를 주며 프로그람언어로서의 상식을 준다.

Perl장의 내용을 파악하기전에 제6장에서 소개된 정규식을 다시 상기하자. 정규식들은 **통용기호**(Wildcard)를 리용하여 사용자가 찾으려는 패턴을 정의한다. 정규식에서 사용자가 찾는 패턴을 정의할 때 "정규식"과 "패턴정합"이라는 용어를 사용한다. 이것은 반드시 이렇게 해야만 하는것은 아니다. 아래에 정규식에 대한 정확한 리해를 준다.

- 정규식들은 쉘에 의해 사용되는 파일정합패턴들에 따라 다르다. 정규식들은 쉘과 Perl을 포함한 많은 다른 프로그람들에서 사용된다. 쉘과 find와 같은 프로그람들에 의해 조화된 파일들은 사용된 정규식과 다르다.

- 정규식은 한개 웃반점안에 기입한다. 이 장에서 **메타기호**(Meta-Character)는 인수처럼 쉘을 통과하기 위하여 인용되여야 한다. 이 장에서는 대부분의 정규식들을 고찰한다.

Perl, sed, awk, vi, grep 와 같은 많은 프로그람을 사용할 때 프로그람을 찾는 정규식이 제시된다. 지령을 리용하면 사용자에게 주는 패턴을 찾을것이다. 패턴은 기호렬처럼 단순하고 통용기호로 쓸수 있다. 많은 프로그람에서 사용된 통용기호들을 메타기호라고 부른다.

Perl 선택항목

Perl 스크립트를 실행하기전에 다시 보아야 할 Perl 과 관련된 많은 선택항목들이 있다. 실례로 사용자는 지령행에서 한개 행 Perl 프로그람을 실행할수도 있고 사용자가 그것을 실행하기전에 더 긴 프로그람들을 쓸수도 있다. 또한 사용자는 지령행에서 문법을 검사할수 있다. 다음 실례는 Perl-h 와 -h 를 실행함으로써 제시된 Linux 체계에서 Perl 의 선택항목들을 보여 준다.

 # perl -h
Usage: perl [switches] [--] [programfile] [arguments]
-o [octal]　　기록분리기호를 지적한다(인수가 없으면 \ 0).
-a　　-n 또는 -p 와 함께 자동구분방식이다($-은 @F 으로 구분).
-c　　문법만 검열한다(BEGIN 과 END 블로크를 실행).
-d[: debugger]　　debugger 부터 실행한다.
-D[number/list]　　추적신호기를 설정한다. (인수는 마스크 혹은 신호기의 비트이다.)
-e'command'　　스크립트의 한개 행이다. [프로그람파일]은 쓰지 않고 몇개의 -e 만 쓸수 있다.
-F/pattern/　　Pattern 은 자동구분방식(-a)을 ()으로 구분한다. //을 선택할수 있다.
-i[extension]　　이 위치에 편집할 내용을 쓴다. (확장을 지적하면 묶어 진다.)
-Idirectory　　@INC/#include 등록부를 지적한다. (한개이상 사용할수 있다.)
-l[[octal]　　행처리를 끝내고 행끝기호를 표시한다.
-[mM][-]module…사용자의 스크립트를 실행하기전에 'use/no module…'을 실행한다.
-n　　사용자의 스크립트에서 순환 'while(< >){…}'을 취한다.
-p　　-n 에서 취한 순환을 취하지만 출력행도 만든다.
-P　　C 전처리기로써 스크립트를 실행한다.
#

사용자는 Perl 프로그람을 쓰기 시작할 때 의심할바없이 마지막에 이 선택항목들의 일부를 사용한다. 사용자프로그람이 더 복잡해 지면 사용자는 프로그람문법을 검사하기 위한 -c 선택항목을 사용하여야 한다.

다음 부분에서는 단순한 Perl 프로그람의 일부를 제시하기로 한다.

입출력파일의 열기, 파일검사연산자, 탈퇴렬

Perl 은 다방면적이므로 임의의 프로그람목적에 사용된다. Perl 에서 대부분 공통적으로 사용하는것들중의 하나는 어떤 방법으로 파일을 읽고 그것을 다루기 위하여 사용하는가 하는것이다. 많이 쓰이는 가까운 실례를 보자.

실례에서는 체계에서 /etc/passwd 파일을 열고 표준출력(STDOUT)로써 한행씩 그 내용을 표시한다. 다음의 실례는 test.pl 이라고 부르는 스크립트를 보여 준다.

```
#！/usr/bin/perl
print  "\n Work with passwd file \n \n";
open（password_file， '/etc/passwd'）||
          die（"passwd file unavailable \n"）;
@line = <password_file>;
print（@line）;
```

이 간단한 프로그람은 Perl 의 많은 중요한 개념들을 보여 준다. 첫번째 행은 Perl 의 경로를 보여 준다. 이 행은 사용하고 있는 Perl 의 종류와 사용하는 UNIX 종류에 의존한다. 그러나 대부분의 경우에/usr/bin/ Perl 에서 Perl 을 찾을것이다. 두번째 행은 프로그람을 실행할 때 화면에 현시된 통보이다. 통보가 현시되기전에 한개의 행바꾸기기호(\n)이 현시되였고 통보가 현시된후에 두개의 행바꾸기기호들이 현시되였다. 다음행은 /etc/passwd 파일을 열고 그것에 passwd-file 의 조종번호를 준다. 만일 파일을 열수 없다면 중단시키고 통보문 passwd file unavailable 를 현시한다. Die 함수는 프로그람을 끝내며 이 장의 뒤의 프로그람에서 사용되는 warn 함수는 사용자가 렬거하는 통보를 출력한다. 다음행은 /etc/passwd 에서 모든 행을 기억시키기 위하여 @행이라는 배렬을 사용한다. 마지막행은 STDOUT 를 위한 @행의 배렬속에서 모든 원소를 출력한다. 변수앞에 처리하는 @는 이것이 배렬이라는것이다.

다음 실례에서 보여 주는것처럼 test.pl 실행결과는 이 경우에 사용자의 말단창문인 STDUT 에 /etc/passwd 의 내용을 보내주는것이다.

```
# ./test.pl

Work with /etc/passwd file on Linux system

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

```
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:102:X Font Server:/etc/Xll/fs:/bin/false
gdm:x:42:42::/home/gdm:/bin/bash
postgres:x:101:233:PostgreSQL      Server:/var/lib/pgsql:/bin/
bash
squid:x:102:234::/var/spool/squid:/dev/null
martyp::500:500::/home/martyp:/bin/bash
hp:x:501:501::/home/hp:/bin/bash
test:x:502:502::/home/test:/bin/bash
#
```

"./"가 test.pl 이름앞에 놓여야 하기때문에 현재 경로에 없는 등록부로부터 이 프로그람을 실행한다는것을 주의해야 한다. 선택적으로 사용자의 경로에 현재등록부를 추가할수 있다. 이 장의 전반에서는 프로그람을 실행할 때 ./을 사용하기로 한다.

STDOUT 보다 파일로 프로그람의 출력을 재지정하는것이 더 좋을수도 있다.

다음실례는 프로그람 test.pl 을 현재등록부에서 passwd.out 로 변경하여 다시 STDOUT 로 보여 준다.

```
# cat test.pl
#
# ! /usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, '/etc/passwd') ||
          die ("passwd_file unavailable \n");
open (STDOUT, ">passwd.out") ||
          die ("couldn't open passwd.out \n");
@line = <passwd _ file>;
print (@line);
# ./test.p1
Work with /etc/passwd file on Linux system
```

```
#cat passwd.out
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:102:X Font Server:/etc/Xll/fs:/bin/false
gdm:x:42:42::/home/gdm:/bin/bash
postgres:x:101:233:PostgreSQL
Server:/var/lib/pgsql:/bin/bash
squid:x:102:234::/var/spool/squid:/dev/null
martyp::500:500::/home/martyp:/bin/bash
hp:x:501:501::/home/hp:/bin/bash
test:x:502:502::/home/test:/bin/bash
#
```

출력에서 첫 부분은 우의 프로그람 test.pl 의 내용을 보여 준다. 다음 부분은 test.pl 의 실행과정을 보여 준다. 다음 행의 "Linux 체계의 /etc/passwd 파일실행중"이라는 통보는 아직 STDOUT 에 있다는것을 알린다. 마지막부분에 표시된것은 test.pl 의 실행결과 나타난 passwd.out 파일의 내용을 보여 준다. 우선 이 프로그람에서 /etc/passwd 파일이 존재하는가를 검사한다. 이것을 할수 없으면 파일을 열어 보고 Die 를 실행한다. 파일에 대하여 많은 검사를 할수 있는데 례컨대 아래의 코드를 통하여 파일이 존재(-e)하는가를 검사할수 있다.

```
if (-e "/etc/passwd") {
        print "/etc/passwd exist \n";
}
else {
        print "/etc/passwd does not exists \n";
}
```

/etc/passwd 에서 리용되는 -e 는 /etc/passwd 가 존재하면 true, 존재하지 않으면 false 값을 취한다. 다음은 Print 행이 실행된다(우의 실례의 해당한 부분에서 if 와 else 를 사용하였다.).

-e 에 추가되는 많은 파일검사연산자가 있다. 표 26-1 에서는 일반적으로 사용하는 파일검사연산자를 보여 준다.

표 26-1 **파일검사연산자**

파일검사연산자	설명 - 다음의것은 항목의 설명이다.
-b	블로크장치이다.
-c	기호장치이다.
-d	등록부이다.
-e	존재한다.
-f	보통파일이다.
-g	설정한다.
-k	비트설명한다.
-l	기호적련결이다.
-o	현재 사용자가 소유한다(UID).
-p	PiPe 이름이다(FIFO).
-r	읽을수 있다.
-s	임의의 통보를 포함한다. 크기는 령이 아니다.
-t	말단을 표시한다.
-u	Setuid 를 가진다.
-w	유용한 UID 와 GID 를 쓸수 있다.
-x	실행할수 있다.
-z	비게 한다.
-A	처리날자를 가진다.
-B	2 진파일이다.
-C	처리날자가 있는 색인마디를 가진다.
-M	변경날자를 가진다.
-O	현재 사용자에 의해 소유된다.
-R	현재 사용자에 의해 읽을수 있다.
-S	소케트이다.
-T	본문파일이다.
-W	현재 사용자에 의해 쓸수 있다.
-X	현재 사용자에 의해 실행될수 있다.

프로그람에는 또한 새로운 행(/n)을 출력하는 행도 있다. 이것은 특수한 문자결합으로서의 탈퇴렬이다. 표 26-2 에서는 perl 에서 일반적으로 쓰는 탈퇴렬을 보여 주었다(이것들은 사용자들의 perl 배치에 의존한것일수 있다.).

표 26-2 탈퇴렬

탈퇴렬	설 명
¥ a	종소리와 소리정보
¥ b	BackSpace 건
¥ cx	조종체 X
¥ e	탈퇴
¥ f	양식만들기
¥ l	다음에 오는 문자를 소문자로 바꾸기
¥ L	다음에 오는 모든 문자를 소문자로 바꾸기
¥ n	새행
¥ r	귀환처리
¥ t	표쪽
¥ u	다음에 오는 문자를 대문자로 바꾸기
¥ U	다음에 오는 모든 문자를 대문자로 바꾸기
¥ V	수직표쪽
¥ xxx	16 진과 8 진값(즉 ¥ 033 8 진은 1X1B16 진)

파일의 읽기와 쓰기를 위하여 파일을 열고 파일검사연산자와 탈퇴렬을 언급하였으므로 이제는 변수에 대하여 보기로 하자. @행은 배렬에 대입할 때에만 변수를 리용할수 있다.

스칼라변수와 배렬변수

기본적인 읽기쓰기조작을 할수 있으므로 파일에 관계되는 자료관리에 대하여 고찰하자.

/etc/passwd 파일을 열고 배렬에 값을 줄 때 이미 배렬에 대한 몇가지 고찰을 하였다. 변수들에 대하여 좀 더 구체적으로 전개해 보자.

먼저 /etc/passwd 파일내용을 주는 배렬을 더 구체적으로 보자. 배렬의 원소개수를 포함하는 $length 라는 스칼라변수에 값주기하는것을 고찰하자. 스칼라변수는 한개 값을 가진다. 이 경우에 $length 의 한개값은 배렬의 원소개수를 포함한다. 아래에서는 $length=@line 행을 포함하고 변수 $length 을 출력하는 변경된 test.pl 프로그람의 내용을 보여 주고 있다.

```
# ! /usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file,  '/etc/passwd') ||
            die ("passwd file unavailable \n");
 open (STDOUT,  ">passwd.out") ||
           die ("couldn't open passwd.out \n");
@ line = <passwd_file>;
```

```
 $length = @line;
 print "The length of the array containing the contents of /etc/passwd is $length \n";
#print (@line);              #설명문행
```

아래에서는 passwd.out 의 내용을 보여 준다. 이것은 test.pl.passwd.out 의 출력이 /etc/passwd 파일에 있는 행의 개수를 포함한다는것을 보여 준다. 또한 /etc/passwd 파일안에 현재 있는 행의 개수를 보기 위하여 wc 지령을 수행시킨다.

```
#cat passwd.out
/etc/passwd 의 내용을 구성하는 배렬의 길이는 23 이다.
#cat/etc/passwd | wc
  23     27     810
#
```

이 출력결과는 23 이라는 값을 가진 test.pl 로부터 얻어 지는 배렬의 길이 ($length) 를 보여 준다. 이것은 wc 지령을 리용해서 확인할수 있는 /etc/passwd 의 행수와 일치한다. wc 출력의 세개의 마당들중 첫번째는 파일의 행수이다. 이것은 /etc/passwd 의 스칼라변수 $length 가 등록된 개수를 포함한다는것을 말해 준다. 스칼라변수들은 여러가지 형을 가진다. 그러나 사용자는 perl 에 변수의 형을 렬거할 필요는 없다. 아래에 스칼라변수의 실례를 아래에서 주었다.

```
$string = "output";
$number=50;
$number=50.77;
$length=@line;     (이것은 앞의 실례에서 리용되였다.)
```

/etc/passwd 안에 23 개 구성요소가 배렬되여 있으므로 배렬에서 한개 요소 혹은 여러개 요소를 선택할수 있다. @line [3] 을 출력하기 위하여 명령문을 쓰는것은 배렬에서 네번째 성분으로 되는(묶음은 @line [0]부터 시작하며 3 은 네번째이다.) /etc/passwd 파일의 네번째 행이다. 아래에서는 test.pl 과 출력파일 passwd.out 를 보여 준다.

```
# cat test.pl
#!/usr/bin/perl
 print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, ' /etc/passwd ') ||
          die ("passwd file unavailable \n");
 open (STDOUT, ">passwd.out") ||
          die ("couldn't open passwd.out \n");
@line = <passwd_file>;
$length = @line;
print "The length of the array containing the contents of etc/passwd is $length \n";
```

```
print $line[3];
#print (@line);            #설명문행출력
```

```
# ./test.pl

Work with /etc/passwd file on Linux system

# cat passwd.out
The length of the array containing the contents of /etc/passwd is 23
adm:x:3:4:adm:/var/adm:
#
```

passwd.out 를 보면 배렬의 길이를 포함하는 명령문이 나타나고 /etc/passwd 안의 네번째행이 련달아 나오는데 그것은 @line［3］에 포함되여 있다.

배렬과 스칼라변수는 소개에서 차이가 있다. 배렬은 @line 으로, 스칼라변수는 $length 로 소개한다. 배렬은 @로 지적하고 스칼라는 $로 지적한다. 스칼라($)와 배렬(@)을 같은 이름도 소개할수 있지만 그것들은 서로 련관이 없다. 다음의 실례에서 line 은 스칼라변수와 배렬이지만 그것들은 서로 련관이 없다는것을 알수 있다.

```
$line = "scalar"
@line = ("line", "of", "text");
print $line          #스칼라변수행포함
print $line[3]       #배렬행의 세번째요소
print @line          #배렬행의 전체 요소들
```

실례에서는 두개의 스칼라 및 배렬이 line 이라는 같은 이름을 가지지만 서로 련관이 없다.

조건명령문과 순환, 연산자, 자동증가와 자동감소

배렬에서 한번이상 순환하는 프로그람을 고찰하자. test.pl 을 0 부터 5 까지 순환하면서 출력파일에 출력하도록 하자.

다음실례에서는 이것을 위한 갱신된 프로그람을 보여 준다.

```
#!/usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, ' /etc/passwd ') ||
        die ("passwd file unavailable \n");
open (STDOUT,  ">passwd.out") ||
        die ("couldn't open passwd.out \n");
@line = <passwd_file>;
$length = @line;
```

```
print "The length of the array containing the contents of etc/passwd is $length \n \n";

$lineno = 0;
            # 배렬에 들어 가기 위한 행번호설정
while ($lineno < 6) {
           #6번 순환한다. (0부터 5까지)
print "This is line number $lineno in /etc/passwd \n";

print $line[$lineno];        # /etc/passwd로부터 행을 현시한다.
$lineno++              # 증가계수기
}
#print (@line);           # 설명문행
```

/etc/passwd의 요소를 여섯번 출력하기 위하여 while 순환을 리용하였다. 아래에서는 passwd.out의 내용을 표시한다.

```
The length of the array containing the contents of /etc/passwd is 23

This is line number 0 in /etc/passwd
root:x:0:0:root:/root:/bin/bash
This is line number 1 in /etc/passwd
bin:x:1:1:bin:/bin:
This is line number 2 in /etc/passwd
daemon:x:2:2:daemon:/sbin:
This is line number 3 in /etc/passwd
adm:x:3:4:adm:/var/adm:
This is line number 4 in /etc/passwd
lp:x:4:7:lp:/var/spool/lpd:
This is line number 5 in /etc/passwd
sync:x:5:0:sync:/sbin:/bin/sync
```

passwd.out는 /etc/passwd의 행번호 0-5(실지는 1-6)을 보여 준다. 실례에서 리용한 while순환을 perl프로그람에서는 다른 조건명령문으로 쓸수 있다.

if-then-else

```
if (식)
{
'then' 명령문블로크
}
else
```

{
'else' 명령문블로크
}

unless-else

unless（식）
{
'unless'명령문블로크
}
else
{
'else'명령문블로크
}

if-then-elsif-else

if（식）
{
'then'명령문블로크
}
elsif'（expression）
{
elsif 명령문블로크
}
else
{
'else'명령문블로크
}

while

while（식）
{
'while'명령문블로크
}

until

until（식）
{

'until'명령문블로크
}

do-while-until

do
{
'do'명령문블로크
}
while（식）

for

for（명령문; 조건식; 반복명령문）
{
'for'명령문블로크
}

foreach

foreach scalar variable（배렬값）
{
'foreach'명령문블로크
}

***next*（skip the rest of the block）**

next;

***last*（temriinates the loop）**

next;

***redo*;（repeats an iteration of the loop）**

redo;

이것은 사용자가 perl 프로그람에서 사용할수 있는 표준적인 조건명령문이다. 마지막으로 갱신된 test.pl 에서 $lineno 변수는 자동증가된다. 여러가지 방법으로 변수를 자동증가, 자동감소시킬수 있다. 아래에서는 $lineno 라는 변수를 리용한 변수의 자동증가, 자동감소의 가장 일반적인 방법을 몇가지 보여 준다.

```
$lineno = 5;          #     $lineno 에 5 을 준다.
$lineno++;            #     $lineno 을 1 만큼 증가시킨다.
$lineno--             #     $lineno 을 1 만큼 감소시킨다.
$lineno+=2            #     $lineno 을 1 만큼 증가시킨다.
$lineno-=2            #     $lineno 을 둘 감소시킨다.
```

실례에서는 사용된 자동증가에 첨가된 많은 연산자들이 있다. 표 26-3 에서는 가장 일반적인 몇가지 연산자를 보여 준다.

표 26-3 자료연산자들

연산자	설 명
+	더하기
-	덜기
*	곱하기
/	나누기
<	보다 작기(수에 대하여)
>	보다 크기(수에 대하여)
= =	같기(수에 대하여)
<=	작거나 같기(수에 대하여)
>=	크거나 같기(수에 대하여)
!=	같지 않기(수에 대하여)
<=>	비교(결과가 크면 1, 작으면 -1, 같으면 0 이다. (수에 대하여))
lt	작기(기호렬에 대하여)
gt	크기(기호렬에 대하여)
le	작거나 같기(기호렬에 대하여)
ge	크거나 같기(기호렬에 대하여)
eq	같기(기호렬에 대하여)
ne	같지 않기(기호렬에 대하여)
cmp	비교(기호렬에 대하여)
=	$a=$a+5 처럼 변수에 값주기
+=	$a+5=5 처럼 증가시켜 더하기
++	$a++처럼 1 만큼 증가
-=	$a-=5 처럼 증가시켜 덜기
--	$a--처럼 1 만큼 감소
/=	$a/=5 처럼 증가나누기
*=	$a*=5 처럼 증가곱하기
**=	$a**=5 처럼 증가제곱
lc 와 \L	기호렬을 소문자로 교체

(표계속)

연산자	설 명
lcfirst 와 \l	기호렬의 첫 문자를 소문자로 교체
uc 와 \U	기호렬을 대문자로 교체
ucfirst 와 \u	기호렬의 첫 문자를 대문자로 교체
||	론리적 or 연산
&&	론리적 and 연산
!	론리적 not 연산
&&	비트별 and 연산
		비트별 or 연산
^	비트별 배타적 or 연산
~	비트별 not 연산
<< >>	왼쪽밀기 및 오른쪽밀기연산
not	론리적 not
and	론리적 and
or	론리적 or
xor	론리적 xor

Perl 에 배치된 기타 몇가지 보충적연산자들도 있지만 이 표의 연산자들만 리용하면 방대한 량의 연산들을 처리할수 있다.

쉘프로그람의 인수

사용자는 $ARGV 라는 특정한 배렬을 리용하여 perl 프로그람인수와 부분루틴의 인수들을 사용할수 있다. Perl 프로그람이름을 입력할 때 현재 사용해야 할 인수도 쉘프로그람에 입력해야 한다.

test.pl 프로그람에 다음과 같은 순환을 포함시키겠다.

```
foreach (@ARGV) {
                print "$_\n"
}
```

인수들이 $_[첨수]라는 배렬속에 있으므로 perl 프로그람에서 $_를 출력할 때 모든 인수들은 STDOUT 에로 보낸다. 다음의 실례에서는 모든 인수들을 출력하게 될 순환을 포함시키도록 변경한 perl 프로그람을 보여 준다. STDOUT 와 관련한 행과 /etc/passwd 의 첫 6 개 인수를 출력하는데 련관된 형이 현재 있다는것을 주의해야 한다. 아래에서는 프로그람의 실행실례를 보여 준다.

```
# cat test.pl

#!/usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, '/etc/passwd') ||
          die ("passwd file unavailable \n");
#open (STDOUT, ">passwd.out") ||
#       die ("couldn't open passwd.out \n");
@line = <passwd_file>;
$length = @line;
print "The length of the array containing the contents of /etc/passwd is $length \n \n";

foreach (@ARGV) {
          print "$_\n"
}

#$lineno = 0;
                              #첫 배렬원소에 행번호를 설정한다.
#while ($lineno < 6) {
                              # 첫 6개수(0-5)를 인쇄하기 위한 while 순환
# print "This is line number $lineno in /etc/passwd \n";
# print $line[$lineno];       # /etc/passwd로부터 행출력
# $lineno++                   # 증가계수기
#}
#print (@line);               # 설명문행

# ./test.pl  a  b  c

Work with /etc/passwd file on Linux system

The length of the array containing the contents of /etc/passwd is 23

a
b
c

#
```

test.pl 을 실행할 때 a b c 를 인수로 입력한다는것을 주의해야 한다. a b c 인수도 foreach 순환을 진행한 결과를 STDOUT 에로 보낸다. 아래의 프로그람에서 사용자는 두개의 인수를 지적하고 있는데 그것은 프로그람이 실행될 때 인수를 보기 위해서가 아니라 프로그람에서 사용될 두개의 인수를 저적하기 위해서이다. 프로그람에 첨가될 다음행에서 보여 주는것처럼 실례에서는 사용자가 새로운 행을 사용할 때까지 인수를 통해 chomp 라는 새로운 행의 제거명령을 리용하고 인수들만 남기고 있다.

```
print "\nEnter the name for which you want to search: " ;
chomp($search = <STDIN>);
print "\nEnter the replacement name: ";
chomp($replace = <STDOUT>);
print "\n You want to search for $search and replace with $replace \n \n";
```

아래에서는 두개의 인수를 요구하도록 갱신된 test.pl 프로그람의 실행의 실례를 보여준다.

```
# cat test.pl
#!/usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, '/etc/passwd') ||
          die ("passwd file unavailable \n");
#open (STDOUT, ">passwd.out") ||
#         die ("couldn't open passwd.out \n");
@line = <passwd_file>;
$length = @line;
print "The length of the array containing the contents of /etc/passwd is $length \n \n";

print "\nEnter the name for which you want to search: ";
chomp($search = <STDIN>);
print "\nEnter the replacement name: ";
chomp($replace = <STDIN>);
print "\n You want to search for $search and replace with $replace \n\n";

#$lineno = 0;
                 # 첫 배렬입력을 위한 행번호설정
#while ($lineno < 6){
                 #첫 6개수(0-5)를 인쇄하기 위한 while 순환
#print "This is line number $lineno in /etc/passwd \n";
#print $line[$lineno];      # /etc/passwd 로부터 행출력
```

```
#$lineno++          # 계수기증가
#}
#print (@line);              # 설명행출력
```

 # ./test.pl

```
Work with /etc/passwd file on Linux system
The length of the array containing the contents of /etc/passwd is 23
Enter the name for which you want to search: test
Enter the replacement name: newtest
You want to search for test and replace with newtest

#
```

지금까지는 Perl 프로그람에 인수를 기입하고 또 그 인수를 지령행에 입력하며 인수를 리용하는 사용자를 호출하는 두가지 기능을 고찰하였다. 이제는 이 인수들을 기입할수 있으므로 아래 항목에서 검색과 교체에 대하여 고찰해 보자. 그다음 우의 프로그람에 검색과 교체기능을 첨가한다.

탐색 및 교체

test.pl 프로그람에 search 문자렬로 탐색하며 replace 행으로 교체하기 위한 행을 포함시키자. 사용자에게는 정규식에 대한 론의와 표를 처리하는데서 탐색과 교체할 필요가 제기될수 있다. Perl 에서 탐색과 교체를 할 때 패턴으로서 정규식을 사용한다. 아래의 실례에서 s 는 교체명령을 표시한다. Perl 프로그람에서 test 문자렬을 newtest 로 교체하려면 다음과 같이 쓰면 된다.

```
s/test/NEWTEST/
```

그러면 test 가 NEWTEST 로 바뀐 결과가 나타난다. 전체를 교체하려면 선택항목 g 를 사용한다. NEWTEST 로 교체될 test 의 우아래 경우를 련결하기 위해 무시하는 경우에는 선택항목 i 를 기입한다.

아래에서는 탐색, 교체에 대한 실례를 주었다. /etc/passwd 파일을 읽고 사용자에게 탐색문자렬과 교체문자렬의 입력을 재촉하고 결과를 STDOUT 로 다음과 같이 보낸다.

 # cat test.pl

```
#!/usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, '/etc/passwd') ||
          die ("passwd file unavailable \n");
#open (STDOUT, ">passwd.out") ||
#         die ("couldn't open passwd.out \n");
@line = <passwd_file>;
```

```
$length = @line;
print "The length of the array containing the contents of etc/passwd is $length \n \n";

print " \nEnter the name for which you want to search:";
chomp ($search = <STDIN>);
print "\n Enter the replacement name:";
chomp ($replace = <STDIN>);
print "\n You want to seach for $search and replace with $replace \n \n";

foreach (@line) {
            s/$search/$replace/ig;
            print "$_";
}

# $lineno = 0;
                  # 첫 배렬입력을 위한 행번호설정
# while ($lineno < 6){
                  #첫 6개수(0-5)를 인쇄하기 위한 while 순환
# print "This is line number $lineno in /etc/passwd \n";
# print $line[$lineno];       # /etc/passwd로부터 행출력
# $lineno++                   # 계수기증가
# }
#print (@line);               # 설명행출력
```

 # . / test.pl

```
Work with /etc/passwd file on Linux system

The length of the array containing the contents of /etc/passwd is 23

Enter the name for which you want to search: test

Enter the replacement name: NEWTEST

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
```

```
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP  User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:102:X  Font  Server:/etc/Xll/fs:/bin/false
gdm:x:42:42::/home/gdm:/bin/bash
postgres:x:101:233:PostgreSQL        Server:/var/lib/pgsql:/bin/bash
squid:x:102:234::/var/spool/squid:/dev/null
martyp::500:500::/home/martyp:/bin/bash
hp:x:501:501::/home/hp:/bin/bash
NEWTEST:x:502:502::/home/NEWTEST:/bin/bash
```

우에서 표시된 마지막행은 초기행 test 가 교체된 NEWTEST 행에 대한 두가지 사건을 포함한다. 탐색과 교체에서 전역적선택항목을 사용하지 않는다면 행에서 첫 test 만 NEWTEST 로 교체된다.

목록연산자

목록관리를 위한 많은 연산자들은 Perl 에 들어 있다. 이 연산자들은 목록에 있는 어떤 배렬을 임의의 방법으로 관리하려고 할 때 쓰인다. 우선 이것들중의 한가지 연산자를 보고 다음에 다른 여러개 연산자를 보기로 한다.

split 연산자는 공통기호렬로 구분되는 기호렬을 가지고 그것들을 련결시킨다. 앞의 실례에서 리용한 /etc/passwd 파일은 기호렬로 구성되는데 ":"으로 구분되여 있다. /etc/passwd 의 전체 행을 취하는 짧은 프로그람을 짜고 split 를 가지고 ":"를 제거한 다음 그것을 한개 행으로 만든다. 아래에서는 기호들을 분리하는 ":"을 제거하는 프로그람 /etc/passwd 의 첫 요소를 보여 주고 프로그람실행결과를 보여 준다.

```
# head -1 /etc/passwd
root:x:0:0:root:/root:/bin/bash
# cat test . pl
# ! / usr / bin / perl
open (passwd_file,  '/etc/passwd')  ||
        die ("passwd file unavailable \n");
```

```
@line = <passwd_file>;
@passwdsplit= split(/:/,    @line[0]);
          print  @passwdsplit;
```

```
# ./test.pl
```

```
rootx00root/root/bin/bash
```

기호렬분리기호 ":"이 없어 진 이 프로그람은 요소목록을 현시한다. Perl 은 이러한 목록연산자들을 많이 가지고 있다. 표 26-4 에서는 Perl 에서 리용할수 있는 여러개 연산자들에 대한 설명을 하였다.

표 26-4 일반적으로 사용되는 목록연산자

연산자	설 명
Join	$문자렬=join("분리기호", @배렬 0); 스칼라행에서 배렬에 대하여 련결한다.
Split	@배렬=split("분리기호", $문자렬); 배렬변수에서 기억된 목록에서 기호렬을 분리한다.
push 및 pop	Push (@배렬 1, 2, 3); @배렬에서 목록으로부터 마지막에 세개항목 1, 2, 3 을 첨가한다. Pop 연산자는 목록의 마지막원소를 제거한다.
Sort	sort 는 ASCII 순서로 증가하도록 목록화한다.
Reverse	목록에서 원소들의 순서를 반대로 만든다.
Shift	목록에서 첫 원소를 제거하고 왼쪽 원소들을 보존한다.
Unshift	목록앞에 원소를 첨가하고 그 나머지 모든것을 오른쪽으로 옮긴다.
Grep	지적한 패턴에 알맞는 목록에서 원소를 취한다.
Splice	배렬에서 토막을 자른다.

표 26-4 에서는 가장 일반적으로 사용하는 목록연산자를 보여 주고 있다. Perl 에서 고찰한것이상의 내용들을 가지고 작업하려고 하는 경우에는 이 연산자들에 대한 보충적인 통보를 보면 된다.

부분루틴

다른 프로그람작성언어에서 쓰인 부분루틴 혹은 함수는 Perl 에서도 쉽게 수행된다. 대표적으로 여러개의 인수를 가지고 부분루틴들을 호출하며 기본프로그람에 그 값을 귀환한다.

출력결과를 써넣으려고 하는 test.pl 프로그람파일이 있는가를 검사하는 부분루틴들을 표시하도록 프로그람을 수정하자. 아래에서는 인수 new_file_name 을 받고 부분 check_file_exists 를 호출하는 test.pl 의 실례를 보여 준다. 이 부분루틴들은 렬거되여 있는 출력파일이 존재할 때 경고통보를 보낸다. Warm 함수는 우리가 렬거한 통보를 출력한다.

#cat test.pl

```
#!/usr/bin/perl
print "\n Work with /etc/passwd file on Linux system \n \n";
open (passwd_file, '/etc/passwd') ||
        die ("passwd file unavailable \n");
@line = <passwd_file>;
$length = @line;
print "The length of the array containing the contents of /etc/passwd is $length \n \n";

print "\nEnter the name for which you want to search: ";
chomp($search = <STDIN>);
print "\nEnter the replacement name: ";
chomp($replace = <STDIN>);
print "\n You want to search for $search and replace with $replace \n\n";

print "\nEnter the name of the file to which you want to write the results: ";
chomp($new_file_name = <STDIN>);
open (STDOUT, ">$new_file_name");

# call subroutine check_file_exists
&check_file_exists($new_file_name);

foreach (@line) {
         s/$search/$replace/ig;
        print "$_";
}
#subroutine to check if output file already exists
sub check_file_exists{
               if (-e $_[0]) {
#       die ("$ _ [0] exists, rerun program with new output file name \n");
        warn ("$_[0] exists, you are going to write over $_[0]\n");
}
}
```

```
# ./test.pl
Work with /etc/passwd file on Linux system
The length of the array containing the contents of /etc/passwd is 23
Enter the name for which you want to search: test
Enter the replacement name: NEWTEST
You want to search for test and replace with NEWTEST
Enter the name of the file to which you want to write the results: passwd.out
passwd.out exists, you are going to write over passwd.out
#
```

warn 함수의 부분인 통보가 화면에 표시되였기때문에 이름이 렬거된 출력파일 passwd.out 의 출력으로부터 그것이 존재한다는것을 알수 있다. 경고신호가 발생하여도 프로그람은 여전히 검색과 교체를 진행한다. Check_file_exists 부분루틴들의 밖에서 소개된 함수 die 는 프로그람을 완료한다.

만일 렬거된 파일이름이 있으면 다른 파일이름을 질문하는것과 같은, 부분루틴들을 만드는것과 같은 많은 기능에 대한 보충적내용들이 있을수 있다. 보충적으로 return 이라는 실마리어가 포함될수 있는데 그 실마리어는 부분루틴들의 값을 기본프로그람에로 귀환시켜 준다.

Perl 에는 우선 정의하는 부분들이 있는데 AUTOLOAD 는 루틴들에 호출된 부분루틴들의 이름을 취하는 부분들이다. BEGIN 은 일반부분을 실행하기전에 실행될 코드를 렬거한다. END 는 일반부분이 끝나기전에 실행될 코드를 렬거한다. 이 부분에서는 Perl 에 대하여 간단히 고찰하였다. Web 싸이트는 Perl 에 대한 보다 좋은 정보를 제공해 준다. Perl 의 중요한 개념을 리용한 간단한 실례프로그람들은 사용자가 Perl 프로그람을 잘 작성하도록 많은 도움을 주었다.

제２７장. X Windows 체계

X Windows 체계의 배경

X Windows 는 체계에 토대한 창문환경이 아니라 망에 토대한 창문환경이다. 왜냐하면 Windows 체계에서 UNIX 체계로 창문을 넘기기 위한 착상이기때문이다.

X Windows 는 콤퓨터망을 통하여 창문화된 사용자대면부를 지원한다. 여기서는 X Windows 봉사기와 의뢰기에 대하여 간단히 언급한다. X Windows 는 정확히 사용자콤퓨터의 창문체계라고는 말할수 없지만 망을 통한 창문체계라고는 말할수 있다.

X Windows 는 장치 혹은 그우에서 실행되는 조작체계에 의존하지 않으며 그 모든것은 봉사기와 의뢰기를 필요로 한다. 봉사기와 의뢰기는 서로 다른 두개의 체계일수도 있고 하나의 체계일수도 있는데 구체적인 설명은 하지 않는다. 봉사기는 화면, 건반, 마우스와 같은 입출력장치를 보장하는 프로그람이다. 의뢰기는 응용프로그람과 같은 지령을 봉사기로부터 접수하는 프로그람이다.

봉사기와 의뢰기는 그 사명에서 크게 차이가 있다. X Windows 봉사기는 사용자의 국부체계(이 장에서는 우리가 사용하는 Windows 체계)이고 X Windows 의뢰기는 봉사기와의 통신을 보장하는 응용프로그람(이 장에서는 체계관리도구와 같은 프로그람을 실행시키는 UNIX 체계)이다. 통속적으로 생각하면 의뢰기는 좀 작은 탁상체계이고 봉사기는 그보다 크고 위력한 체계라고 볼수 있다. X Windows 에 대하여 말한다면 그것이 X Windows 를 조종하는 체계이지만 지령에 대답하는 체계로서의 의뢰기이다. 이러한 특성으로 하여 의뢰기를 호스트라고 부르고 있다.

한개 화면을 관리하는 X Windows 에서 건반이나 마우스를 관리하는 프로그람은 X 봉사기로 알려 져 있다. 의뢰기는 X 봉사기에 현시하는 프로그람이다. X 의뢰기는 정보요청을 비롯한 요구를 X 봉사기에 보낸다. X 봉사기는 여러대의 의뢰기들로부터 요청을 접수하여 정보나 오유통보를 귀환한다.

망에서의 Windows 체계우에서는 몇개의 UNIX 호스트의 X 창문을 열수 있다. 심지어 UNIX 체계창문 1, UNIX 체계창문 2, …도 열수 있다.

X 봉사기는 다음과 같은 기능을 수행할수 있다.

- 제기된 요청을 화면에 표시
- 정보요청에 대한 대답
- 요청과 관련한 오유통보
- 건반, 마우스, 화면관리
- 창문의 창조 및 설계, 제거

X 의뢰기는 다음과 같은 기능을 수행할수 있다.

- 봉사기에로 요청을 전송
- 봉사기로부터 사건 및 오유를 접수

X 봉사기쏘프트웨어

많은 X 봉사기들이 만들어 져 상품으로 보급되고 있다. Windows 의 UNIX 환경에서 X Windows 가 어떻게 리용되는가를 설명하기 위하여 Exceed 6(앞으로 간단히 Exceed 로 표시한다.)을 고찰하기로 한다. 이 장에서는 실례로서 Exceed 를 가지고 고찰하기로 한다. 그림 27-1 에 X Windows 의 Exceed 안내구조를 제시하였다.

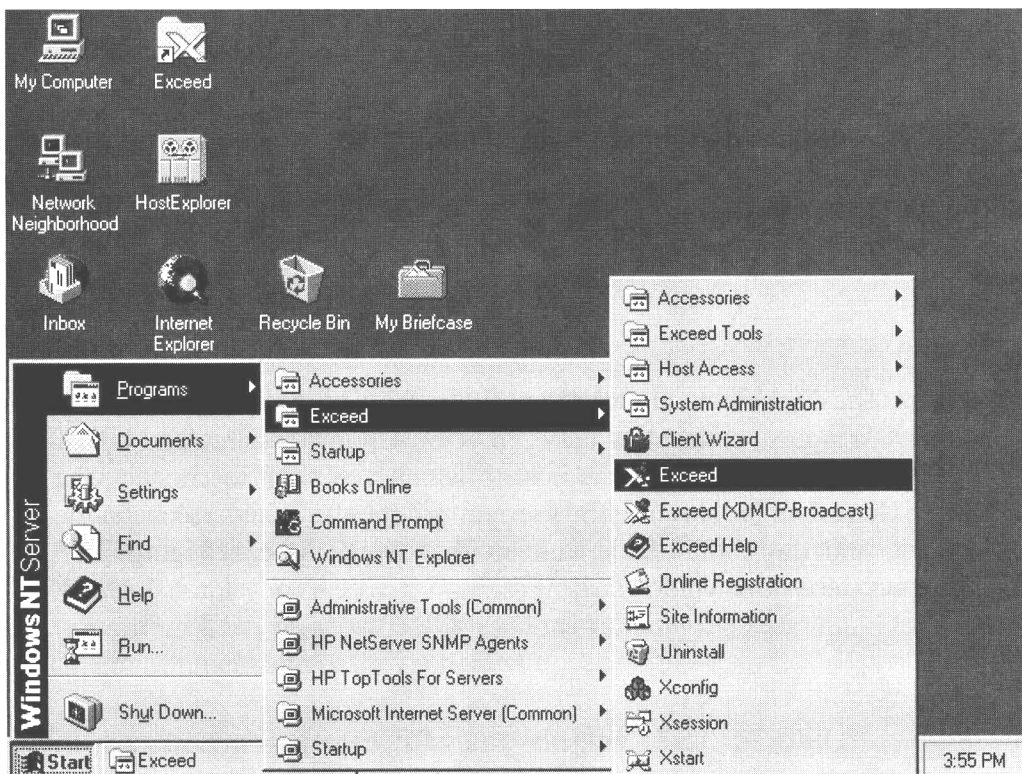



그림 27-1. Exceed 프로그람의 차림표

그림 27-1 의 제일 꼭대기 Exceed 의 그림기호를 선택하여 Exceed 의 그룹을 현시한다. 그림 27-2 에 Exceed 그룹의 그림기호를 제시하였다.

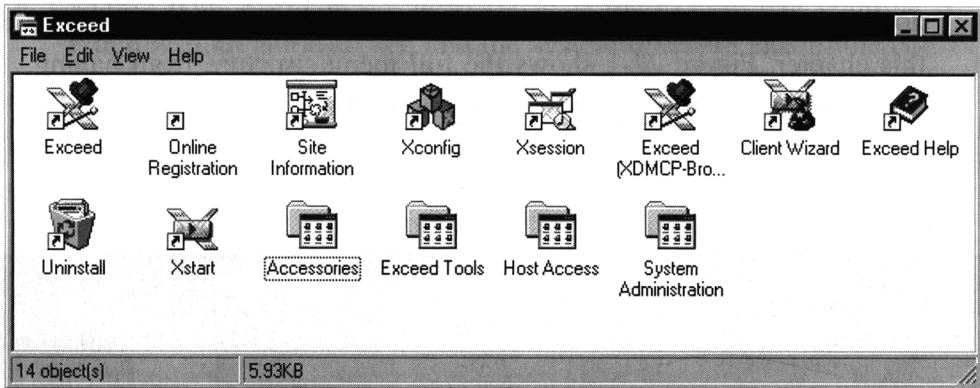



그림 27-2. Exceed 그룹

그림 27-1 의 Exceed 안내를 선택하면 Windows 체계와 UNIX 의 련관 X Windows 를 만들수 있다. 이 경우에 UNIX 체계는 련결하기 위한 호스트, 호스트에 련결하려고 하는 사용자, UNIX 에서 실행하기 위한 지령들을 렬거할수 있다. 그림 27-3 은 X 시작창문을 보여 준다.

DTTERM.XS - Xstart Client Startup Application
File Edit Run! Install... Help
Start Method
RLOGIN (TCP/IP)
Program Type
X Window
Settings
Program...
Other...
Login
User ID: root
Host: 15.32.165.11
Password:
Host Type: HP
Command: dtterm -background white -display 15.32.165.10:0
Prompt: None Login Info Account Info
Description: open dtterm
Show On Client Startup Menu

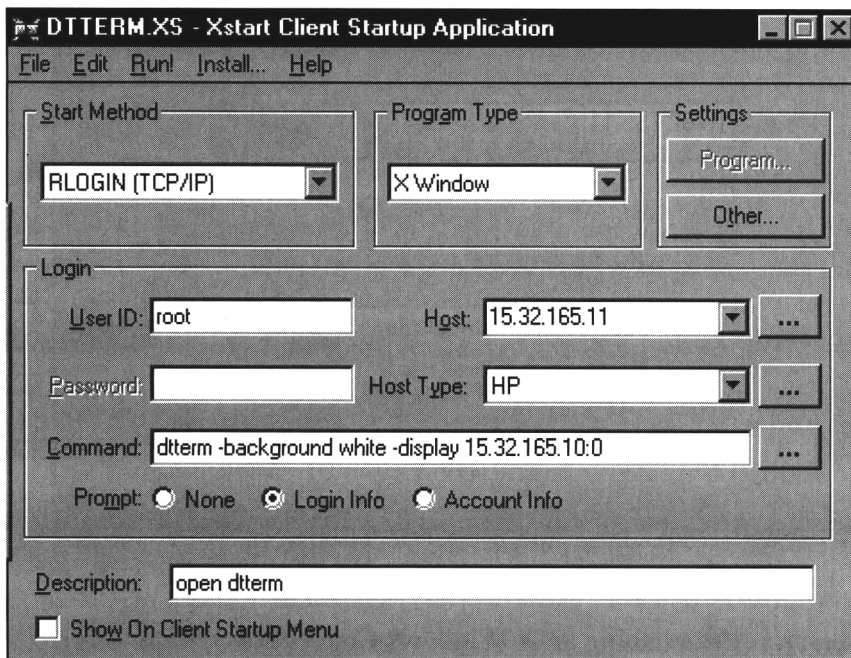



그림 27-3. X Windows 의 설정

그림의 창문제목은 "DTTERM.XS"이다. 요구하는 정보와 함께 X 시작창문을 설치한 후 사용자는 그 구성을 보존할수 있다. 이 경우 지령 dtterm 을 리용하여 그 창문을 자기의 이름으로 보관한다. 체계의 형태는 X Windows 를 실행시킬수 있는 어떤 체계와 류사하다. 앞으로 실례에서는 HP-UX 체계를 사용하는데 여기서 Host Type 는 HP 이다. 대부

분 UNIX 체계우에서 공동탁상환경을 수행하는 완성된 지령 dtterm 을 사용하는것을 아래에서 제시하였다.

dtterm -background white -display 15.32.165.10:0

그림 27-3 의 창문에서 Run!을 선택하면 통과암호를 주거나 기타것을 변경할수 있는 그림 27-4 와 같은 창문이 나타난다.

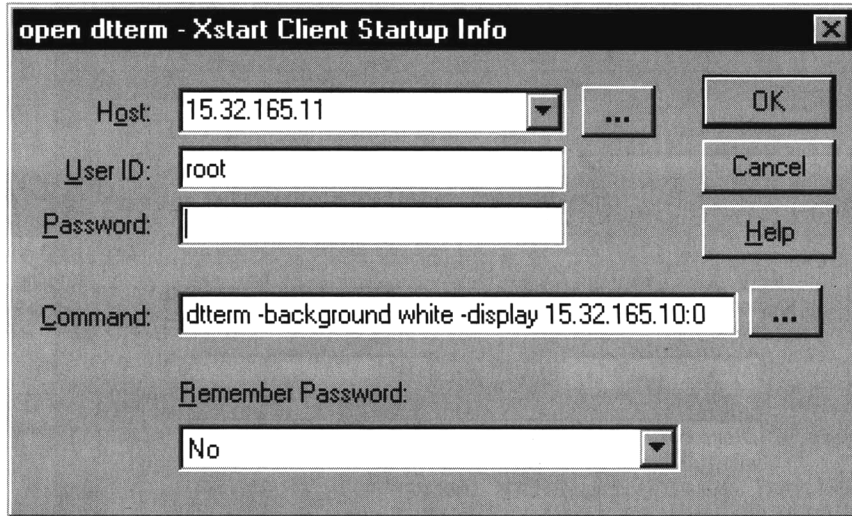



그림 27-4. X Windows 련계를 설명

이 지령도 dtterm 창문을 보여 주는데 그것은 UNIX 의 백색배경을 가진 표준창문프로그람으로서 IP 주소가 15.32.165.11 인 창문을 현시한다. 이 경우에 IP 주소는 사용자가 지령을 쓴 Windows 체계로서의 X Windows 봉사기이다. 여기서 ":0"이라는것은 dtterm 을 위해 사용될 Windows 체계상의 첫 화면이라는것을 의미하는데 그 리유는 X Windows 환경에서는 여러개의 화면을 태울수 있기때문이다. 이 지령이 실행되는 체계는 15.32.165.10 이다. 이것은 X Windows 의뢰기처럼 작용하는 UNIX 체계이다.

사용자는 Windows 체계에 이 정보를 입구한다고 하여도 사용자가 **X 시작칸**(X Start Box)에서 렬거한 UNIX 체계에로 전송된다. 이 전송은 UNIX 체계에서 직접 보이는 dtterm 지령을 입력하는것과 같은 결과를 가져 온다.

통과암호를 입력하고 OK 를 누르면 UNIX 체계로 들어 가기 위한 창문인 dtterm 창문이 Windows 체계우에 나타난다.

그림 27-5 에서는 Windows 체계우에 표시된 dtterm 창문을 보여 준다. 창문은 이때 dtterm 을 위한 HP-UX 안내페지를 현시한다. 사용자가 직접 UNIX 체계상에서 조작한다면 dtterm 창문우에서 어떤 명령이든지 렬거할수 있다.

그림 27-5. UNIX 체계에서 실행되고 Windows 우에서 표시되는 dtterm

그림 27-6. UNIX 체계에서 실행되고 Windows 우에서 표시되는 xterm

X 시작창문을 리용하면 UNIX 체계우에서 적용할수 있는 어떤 프로그람도 실행할수 있다. 그림 27-6 에서는 UNIX 체계에서 실행되고 Windows 체계우에서 현시되는 한개 Xterm 창문을 보여 준다.

이런 환경에서 X Windows 에서는 dtterm 혹은 Xterm 와 같은 종결창문만을 실행시킬수 있다. 사용자는 늘 쓰는 응용프로그람을 실행시키거나 체계기능을 수행시킨다. 그림 27-7 에 체계관리자(SAM)를 보여 주었는데 그것은 HP-UX 의 본래 체계관리도구로서 UNIX 체계우에서 실행되고 Windows 체계에서 핵심부모형을 선택하는데 대하여 표시한다.

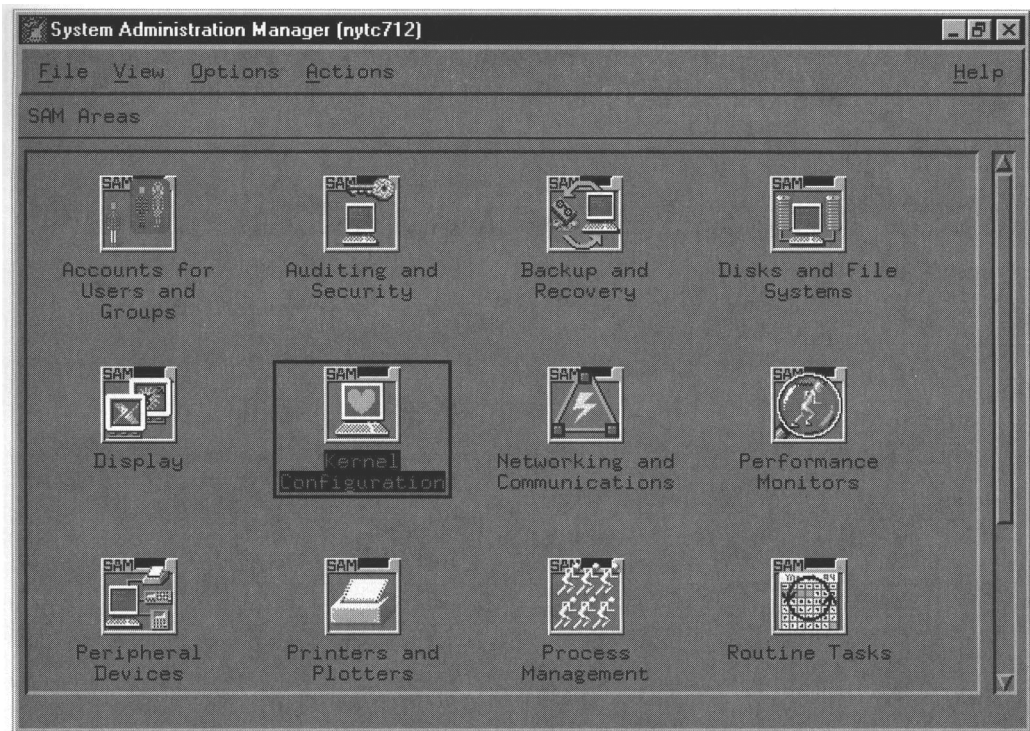



그림 27-7. HP-UX 체계에서 실행되고 Windows 우에서 표시되는 SAM

사용자는 단순한 창문이나 단순한 응용프로그람을 열기 위하여 Exceed 에서 사용할수 있는 한계에 대하여서는 고려하지 않는다.

Exceed 는 또한 UNIX 체계상에서 공동탁상환경(CDE)를 실행시키기 위해서도 리용된다. CDE 는 사용자가 몇개의 탁상화면을 펼칠수 있게 하는 창문환경으로서 여러개 창문으로 나타난다.

Exceed 에서 파라메터에 CDE 환경을 사용자의 Windows 체계상에서 실행하는것을 렬거할수 있다. CDE 는 사용자가 과제를 합리적으로 수행할수 있는 여러개의 작업공간을 가질수 있다. 그림 27-8 은 CDE 가 선택된 작업공간에서 첫번째 작업공간을 호출하여 실행하는것을 보여 준다.

프로그람은 그림 27-8 의 아래의 중간부분에 표시된 4 개 작업공간중의 첫번째것이다. 다른 3 개의 작업공간은 sam, files, icon 이라고 표시된것들이다.

그림 27-9는 체계관리자를 시동할 때 작업공간 sam을 보여 준다.

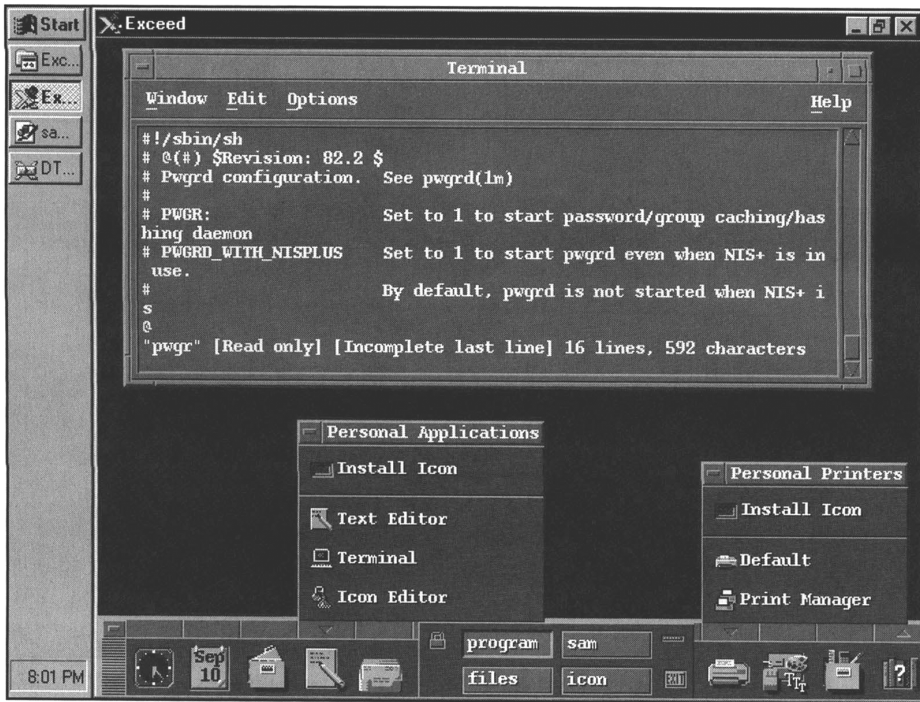



그림 27-8. CDE 프로그람작업공간

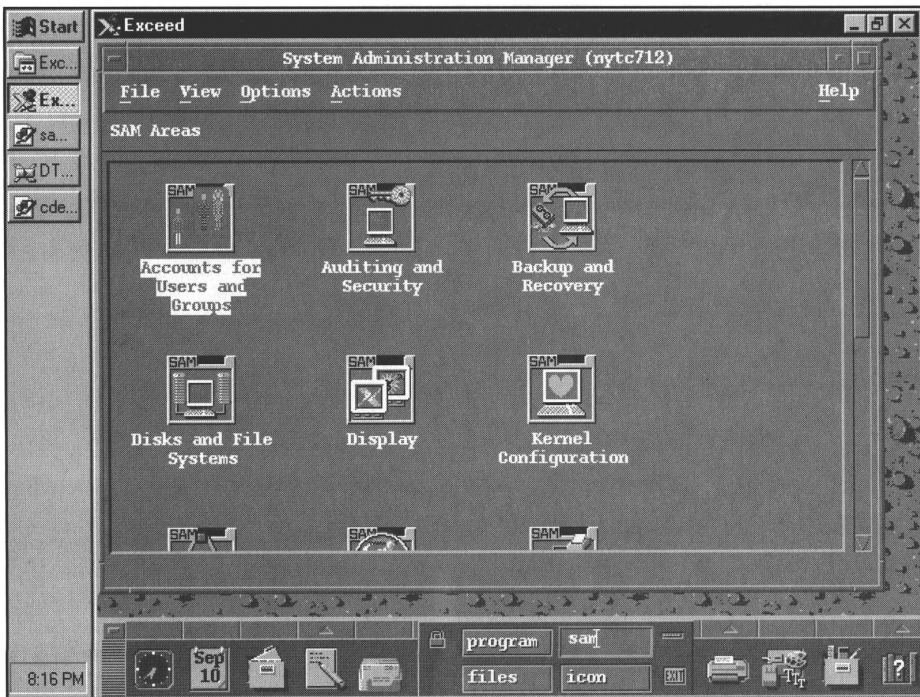



그림 27-9. CDE Sum 작업공간

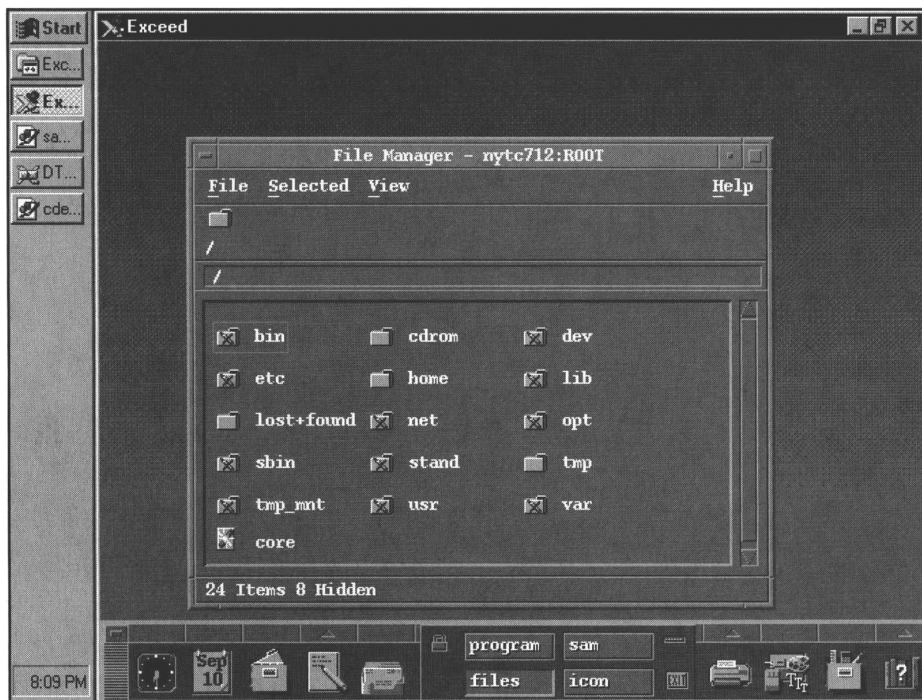



그림 27-10. CDE files 작업공간

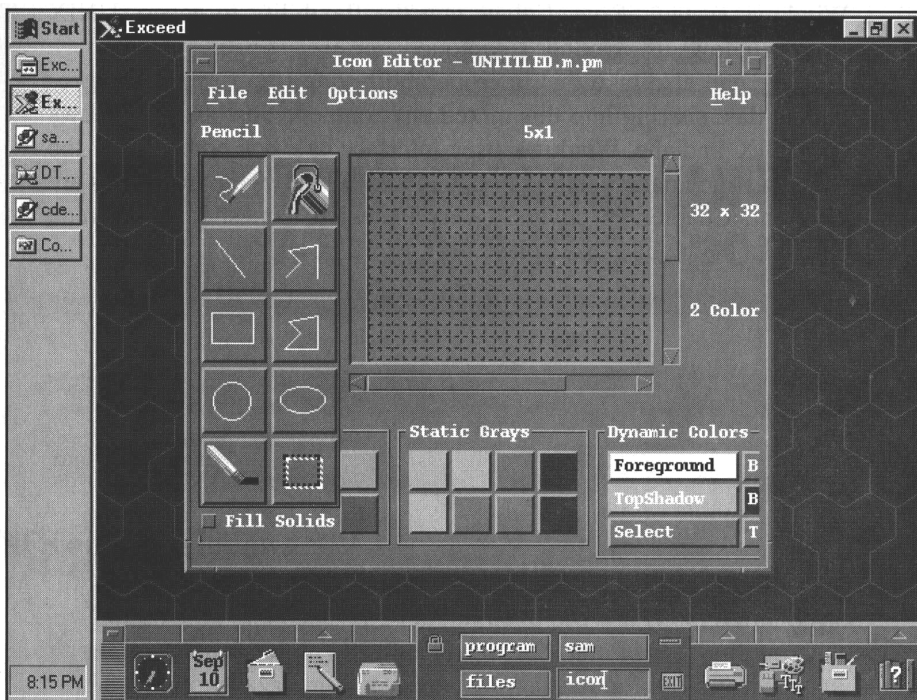



그림 27-11. CDE icon 작업공간

그림 27-10 에서는 files 작업공간을 보여 주었는데 이 작업공간에서는 CDE 파일관리자를 실행시킬수 있다.

그림 27-11 에서는 icon 작업공간을 보여 주었는데 작업공간에서는 그림기호편집기를 기동시킬수 있다.

Windows 체계우에서 UNIX 프로그람을 현시하는 X Windows 를 리용한 이 기술은 이런 다양한 환경속에서 매우 위력한데 이것을 설치하는데는 많은 품이 들지 않고 간단하다. Exceed 를 사용하면 Windows 에서 실행되는 프로그람을 UNIX 에서 열수 있으며 Windows 체계에서 완전한 CDE 를 실행시킬수 있다. UNIX 보다도 Windows 를 자주 리용하는 사용자들은 Windows 에서 dtterm 이나 Xterm 창문을 자주 열어 보아야 한다.

Exceed 의 두가지 기술을 리용하면 Windows 에서 UNIX 를 호출하는 기능을 알수 있다. 바로 이러한 UNIX 혹은 Windows 상의 X Windows 사용자대면부는 적합치 못하며 UNIX 와 Windows 혼잡환경에서의 자료공유는 더우기 차이난다. 제 16 장에서도 UNIX 와 Windows 의 자원공유에 대하여 서술하였다.

제２８장. 망 - UNIX 체계와 Windows 호상작용성

NFS 와 X Windows

이 장은 내부조종과 관련하여 24 장과 서로 련결되여 있다. 이 장에서는 망파일체계(NFS)를 사용하는 UNIX 체계에서 자료를 쉽게 처리하는 Windows 체계의 망처리에 대하여 서술한다. 24 장에서는 UNIX 에서 도형적처리를 보장해 주는 Windows 체계상의 X 봉사기프로그람의 실행에 의해 Windows 와 UNIX 의 내부조종을 일관시켰다. Xwindow 체계(XWindows)를 리용하여 사용자는 Windows 와 UNIX 체계를 련결하는 도형에 대한 표상을 가진다. 또한 NFS 를 리용하여 이 두개의 체계에서 자료공유를 간단히 할수 있는 수단을 가진다. 이 두가지 기술과 XWindows 의 NFS 는 두개의 조작체계사이의 내부조종기능을 지원한다. 이 책의 앞 부분에서 TCP/IP 환경에 대하여 서술하였지만 그것을 여기서 다시 상기시킨다.

TCP/IP 망의 배경

그림 28-1 에서 ISO/OSI 모형의 7 가지 망기능층을 볼수 있다. 사용자가 이 망수준에 대해 파악하도록 하기 위하여 이것을 각이한 측면에서 취급한다. 제일 웃층은 제일 많이 사용하게 되는 층인데 그것은 사용자와 관계되는 기능들이기때문이다. 아래층은 망기능수행을 발전시키기 위한 구성상요구를 실현하고 체계의 전반적수행에 큰 영향을 주는 층이다.

그림 28-1 의 맨아래 1 층부터 시작하여 4 개의 층을 각각 설명한다.

층번호	층이름	자료모양	설 명
7	응용프로그람층		사용자응용프로그람을 사용한다.
6	표현층		응용프로그람이 준비된다.
5	대화층		응용프로그람이 준비된다.
4	전송층	packet	TCP 에 의하여 조종되는 포구에서 포구에로 전송한다.
3	망층	datagram	인터네트규약(IP)은 목적지 또는 암시적인 경로로 가는 경로를 조종한다.
2	련결층	frame	원천 또는 목적지주소로 Ethernet 와 IEEE802.3 에서 요약한다.
1	물리층		체계사이에 물리적련결을 한다. 일반적으로 동축케블과 꼬임쌍선

그림 28-1. ISO/OSI 망기능층

이 모형은 국제표준화기구의 열린체계호상련결모형이다. 이것도 망층의 호상작용을 리해하는데서 도움이 될수 있다.

물리층

콤퓨터에서 작업시작은 체계호상간 물리적련결이다. 물리적층이 없이는 체계들사이에 통신을 실현할수 없다. 물리적층은 케블을 통하여 분석신호로 전송할수 있는 자료를 변환한다(현재 케블을 통한 어떤 물리적층을 가지고 있다고 가정한다.). 통신정보전송은 케블을 벗어나 다음층에서 리용될 준비를 한다.

련결층

사용자체계에 국부적으로 다른 체계를 련결시키기 위하여 국소부위에 다른 모든 체계들을 련결할수 있는 련결층을 리용한다. 이 층은 IEEE802.3 혹은 Ethernet 를 가지고 있는 층이다. "encapsulation"이라는 방법이 있는데 이 방법은 자료가 두가지 형태(IEEE802.3 과 Ethernet)가운데서 하나를 가지기때문에 붙인 이름이다. 자료는 물리적층에 원천과 목적주소 기타 정보수속을 포함한 프레임(자료의 다른 이름)을 통해 전송된다. 이 두 방법은 서로 다르나 IEEE802.3 과 Ethernet 는 서로 밀착되였다. 그래서 두개 층을 가지고 원천과 목적주소를 가진 두 형식을 한개 형식으로 자료와 체계의 련관을 보장한다. 그림 28−2에서는 Ethernet의 구성성분과 IEEE802.3에 대한 해설을 주고 있다.

목적주소	6byte	자료가 가닿을 주소
원천주소	6 byte	자료가 떠나온 주소
형	2 byte	802.3에서 길이계산
자료	46-1500 byte	802.3의 38-1492 byte
crc	4 byte	오유를 찾기 위한 검열합

그림 28−2. Ethernet의 요약

주의해야 할것은 IEEE802.3과 Ethernet의 최대자료크기가 1492byte와 1500byte로 차이난다는 점이다. 이것이 최대전송단위(MTU)이다. 이 자료를 Ethernet 에서 **프레임**(Frame)이라고 부른다(다음층의 자료의 재구성은 IP 에서 *datagram* 이라고 부른다. 그리고 두개 층의 encapsulation 을 TCP 의 파케트라고 부른다.). Ethernet 와 IEEE802.3 은 같은 물리적련결우에서 실행할수 있지만 사실 두 방법사이에는 차이가 있다.

망층

다음은 세번째 층인으로서 이것이 망층이다. 망층은 인터네트규약(IP)과 같다. 이 층에서의 자료를 datagram 이라고 부른다. 이 층은 망을 통하여 자료를 조종하는 층이다.

IP 에 의해 조종된 자료는 때때로 일부 형태의 오유를 주는데 인터네트조종통보규약(ICMP)에 의해 원천체계에 전달된다. IP 가 리용되여도 정보는 Ethernet 에서 충분하지 못하다. 그러한것으로 하여 사용자는 자료토막을 끌낸다. 이것은 자료를 재구성함으로써 사용자가 여러 층들에서 작업할 때 많은 무효과를 방지하게 한다. IP 는 단순한 류형의 망층을 조절한다. 사용자의 체계에 직렬된 목표지에 자료를 보낸다면 자료는 그 체계로 보내 진다. 다른 방식으로 목적지가 사용자의 체계에 직렬되지 않는다면 자료는 다른 기정적통보로 넘어 간다. 때로는 접근방식이라는 기정적통로는 자료를 자기 목적지에 보내게 된다.

전송층

이 층은 망층보다 한계단 높은 층으로서 포구를 통한 통신을 보장한다. TCP 는 이 수준에서 찾은 가장 공통적인 규약으로서 포구에서 포구에로 파케트형태로 전송한다. 이 포구들은 telnet, rlogin, ftp 등과 같은 망프로그람들에서 쓴다. 사용자는 포구들과 련관되여 있는 이 프로그람들이 층일람표에서 보고 알수 있는바와 같이 가장 높은 수준에 있다는것을 알수 있다.

인터네트규약(IP)주소화

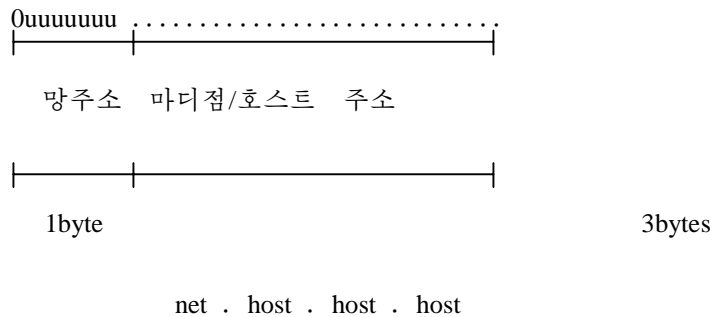
인터네트규약주소(IP 주소)는 클라스 "A", "B", "C"주소이다(클라스 "D", "E"주소도 있지만 여기서는 취급하지 않는다.). 클라스 "A"망은 매개 말단 "B"나 "C"에 비하여 많은 마디점을 가진다. IP 주소를 매개의 마당으로 구분하여 보는것은 마디점(혹은 호스트)주소와 망주소를 정의하기 위해서이다. 그림 28-3 은 클라스와 주소사이의 련계를 요약해서 보여 준다.

클라스주소	망	망마디점	정의하는 망 비트	망의 마디점을 정의하는 비트
A	조금 있다	최대이다	8bit	24bit
B	많다	많다	16bit	16bit
C	최대이다	조금 있다	24bit	8bit
예약	-	-	-	-

그림 28-3. 인터네트규약(IP)주소들의 비교

그림에서 비트는 망의 범위와 매 클라스의 마디점을 규정하는 비트수를 의미한다. 실례로 A 클라스주소는 망을 규정하는데 8bit 가 요구되고 C 클라스주소는 24bit 가 요구된다는것이다. A 클라스주소는 C 보다 적은 수의 망을 보장한다. A 에서는 C 보다 매개 망의 많은 마디점을 지원한다. 그림 28-4 에는 이 주소클라스와 련관된 개별적인 파라메터를 보여 준다.

주소클라스	망지원	망의 마디점	주 소 분 류		
A	127	16777215	0.0.0.1	-	127.255.255.254
B	16383	65535	128.0.0.1	-	191.255.255.254
C	2097157	255	192.0.0.1	-	223.255.254.254
예약	-	-	224.0.0.0	-	255.255.255.255
2 진형식으로 32bit 주소를 보고 주소클라스를 어떻게 결정하는가를 볼수 있다.					

net . host . host . host

클라스 "A"주소는 첫 비트를 0 으로 설정한다. 사용자는 많은 마디점과 호스트주소에 따라 비트로 지원되는가를 볼수 있다. 클라스 A 의 첫 비트는 0 이고 망정의에 사용되는 망부분의 7bit 는 보존된다. 다음 3byte 는 망의 마디점을 정의한다.

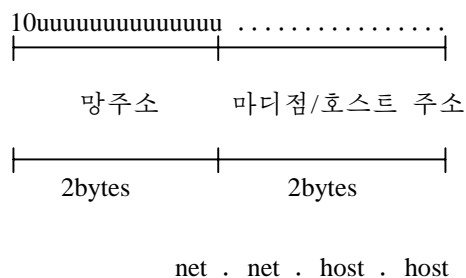
net . net . host . host

클라스 "B"주소는 첫째 비트를 1 로 설정하고 둘째 비트는 0 으로 설정된다. 일부 마디점과 망을 제외하고 많은 망들은 클라스 A 의 주소와 같이 지원한다. 클라스 B 의 주소로는 2byte 를 망부분에 배당하고 2byte 는 주소에 대한 마디점부분에 할당된다.

그림 28-4. 클라스의 주소

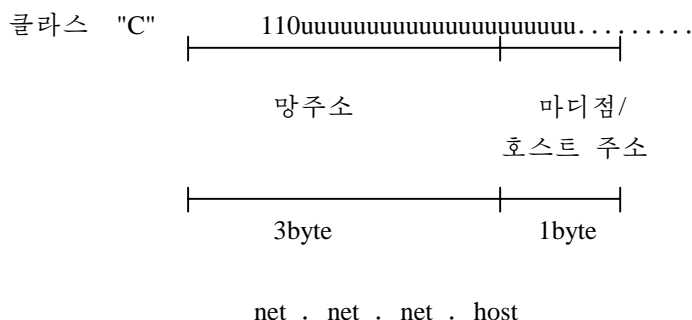
net . net . net . host

클라스 "C"의 주소는 첫째 비트와 둘째 비트에 1 을 설정하고 셋째 비트에는 0 을 설정한다. 망과 일부 망의 마디점에 대한 표시를 큰 수로 하면 클라스 C 의 주소에서 효과적이다. 클라스 C 주소에는 3byte 의 망과 1byte 의 망마디점이 더 있다.

그림 28-4. 클라스의 주소(계속)

사용자망에서 매 련결부는 개별적인 IP 주소를 가진다. 두개 망련결부를 가지는 체계들은 두개의 개별적인 IP 주소를 가져야 한다.

NFS 환경

NFS 에 대한 론의와 실례들은 이 장에서 제한하지 않는다. 파일전송규약(FTP)과 같은 효과적인 파일공유에 대한 봉사들이 있는데 이것은 실례를 통하여 보기로 한다. NFS 는 UNIX 사용자들에게 널리 리용되기때문에 Windows 와 UNIX 환경에서 NFS 를 어떻게 리용되는가를 사용자에게 보여 주려고 한다.

NFS 는 사용자체계에 국부적인것처럼 보이게 하기 위하여 사용자가 원격체계에 디스크를 태우게 한다. 류사하게 NFS 는 원격체계에 대하여 국부적인것처럼 보이도록 사용자 국부디스크를 태우게 한다.

X Windows 와 같은 NFS 에는 고유한 개념들이 있다. 아래에 NFS 의 중요항목의 정의를 보여 준다.

Node	결합되여 있는 콤퓨터체계 혹은 콤퓨터망의 부분이다.
Client	자료들을 요구하거나 다른 마디점(봉사기)으로부터 봉사한다.
Server	자료를 제공하는 마디점 혹은 망의 다른 마디점(의뢰기)에 대한 봉사를 한다.
File System	디스크구획이나 론리적표식이다. 또는 워크스테이션인 경우에 이것은 디스크이다.

Export	NFS 를 리용하는 원격마디점을 태우기 위해 허용하는 파일체계로 만들기 위한것이다.
Mount	NFS 를 리용하는 원격파일체계를 허용하기 위한것이다.
Mount Point	NFS 가 설치된 파일체계우의 등록부이름이다.
Import	원격파일체계를 태우기 위한것이다.

NFS 를 리용하여 어떤 자료가 공유되기전에 UNIX 체계는 보내 진 파일체계로 설치되여야 한다. /etc/exports 파일은 파일체계를 어떻게 보내겠는가를 정의하기 위하여 사용된다.

이 파일은 보내는 등록부, 읽기전용 "ro"와 사용자가 이름이 없이 호출할 때 요구를 조절하는 "anon"과 같은것을 선택할수 있다.

아래에 이름 없는 사용자가 아닌 어떤 사용자들에게 보내 진 /opt/app1 안에 있는 /etc/exports 파일의 실례를 보여 준다.

```
/opt/app1          -anon=65534
/opt/app2          -access=system2
```

사용자 UNIX 체계에 파일을 추가하면 exports ?a 와 같은 프로그람을 실행해도 된다.

이 장에서는 Windows 체계에 의해 설치되여 보내지는 UNIX 파일체계에 대하여 중점을 두고 고찰한다. UNIX 체계에 국부적으로 설치된 원격파일체계는 자주 /etc/fstab 에 출구된다. 아래에 원격파일체계에 국부적으로 설치된 /etc/fstab 에로 들어 가기 위한 실례를 보여 준다. 등록부 system2 우의 /opt/app3 은 /opt/opt3 에서 국부적으로 설치된다.

```
system2:/opt/app3  /opt/app3  nfs  rw, suid  0  0
```

많은 UNIX 체계에서 리용되는 showmount 지령을 리용하여 국부파일체계에 설치 된 모든 원격체계(의뢰기)를 볼수 있다. showmount 지령은 의뢰기에 의하여 NFS 로 자주 설치된 파일체계를 확정하는데 리용된다. showmount 의 출구결과는 호스트이름과 의뢰기가 설치된 등록부를 보여 주기때문에 특별히 읽기 쉽다. 다음과 같은 showmount 의 세가지 선택을 할수 있다.

-a	"이름 : 등록부"형식으로 출구한다.
-d	의뢰기에 의하여 태운 원격인 모든 국부적등록을 표시한다.
-e	보내 진 파일체계목록을 출구한다.

이 장에서는 Windows 상에서 NFS Maestro 를 사용하여 망호상조종가능성을 고찰한다.

사용자는 대표적으로 UNIX 체계에 파일체계를 Windows 우에 설치하기 위해 NFS Maestro 와 같은 NFS 의뢰기를 실행시킬수 있다. 이 설치는 모든 Windows 봉사기들이 NFS Maestro 를 실행시킨다는것을 의미한다. 사용자가 소유하고 있는 Windows 체계의 개수에 의존되므로 사용자는 과제로 될 NFS 적재를 모든 체계에서 찾아야 할 때가 있다. 파일체계는 매개 체계에 NFS 의뢰기를 설치하는것보다 Windows 와 UNIX 사이에서 대문과 같이 작용하는 Windows 체계를 사용할수 있다. NFS Maestro 대문은 사용자 Windows 망을 사용자 UNIX 망에로 다리를 놓는다. 모든 Windows 의뢰기는 Windows 체계에서의 설치의 간단화, NFS 조직으로 하여 UNIX 체계에로의 NFS 호출을 실현하기 위한 NFS Maestro 접근방식을 취하게 된다.

NFS Maestro 접근방식은 대리로 활동하면서 Microsoft 봉사기통보블로크(SMB)망을 UNIX 망에 련결한다. 그것은 Windows 의뢰기의 SMB 요구를 UNIX NFS 봉사기에 주는것이다.

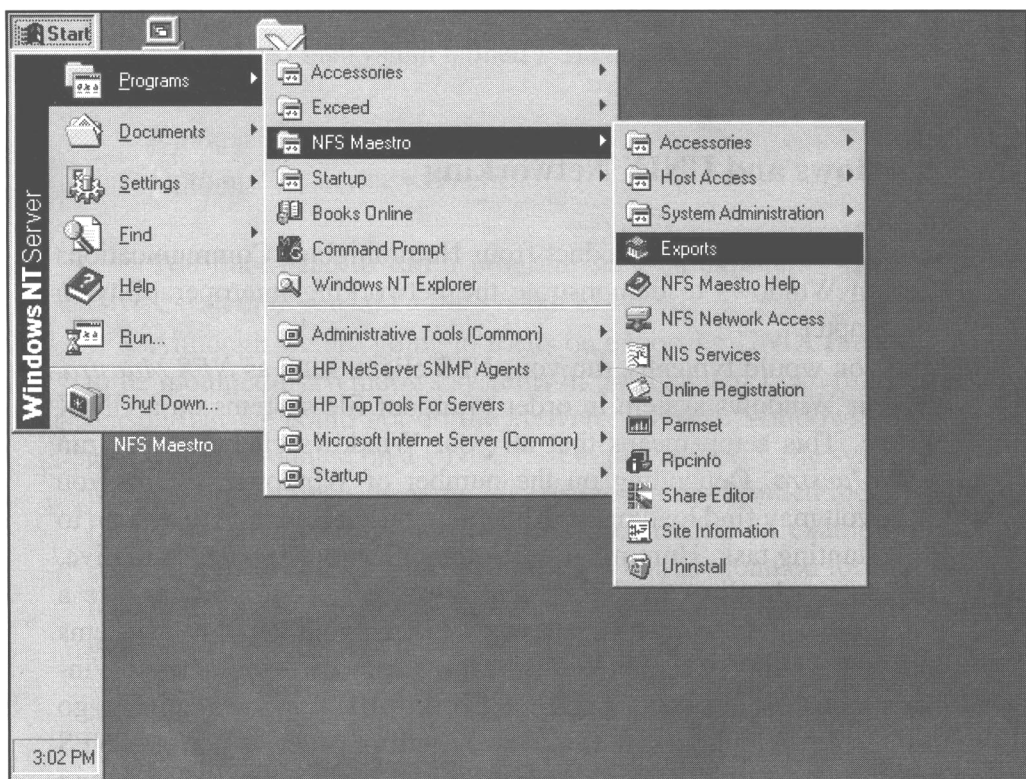



그림 28-5. Windows 에서 표시된 Maestro 안내

제시된 NFS Maestro 의뢰기를 사용하는것은 NFS Maestro 접근방식을 사용하는것보다 좋다. 많은 체계관리원칙들과 마찬가지로 간단화와 실현사이의 교환은 금지되여 있다. 이 장에서는 UNIX 봉사기우에서 파일체계호출을 위한 NFS Maestro 의뢰기사용에 대하여 설명한다.

그림 28-5 는 설치된 Maestro 처리안내를 보여 준다.

그림 28-5 에서 보는것처럼 NFS 기능보다 훨씬 좋은것들이 NFS Maestro 에 있다. 이 장을 취급하는 목적은 망과 관련된 Windows 와 UNIX 사이의 중요한 조종기능원칙은 고찰하는것인데 이 장의 뒤부분에서 몇가지 보충기능을 더 취급하기로 한다.

NFS 그림기호들은 NFS Maestro 그림기호그룹의 한 부분으로서 그림 28-6 에서처럼 호출될수 있다.

NFS Maestro

File Edit View Help

Name	Size	Type	Modified
Accessories		File Folder	8/15/13 2:00 PM
Host Access		File Folder	8/15/13 2:00 PM
System Administration		File Folder	8/15/13 2:00 PM
Exports	1KB	Shortcut	8/15/13 2:28 PM
NFS Maestro Help	1KB	Shortcut	8/15/13 2:28 PM
NFS Network Access	1KB	Shortcut	8/15/13 2:28 PM
NIS Services	1KB	Shortcut	8/15/13 2:28 PM
Online Registration	1KB	Shortcut	8/15/13 2:28 PM
Parmset	1KB	Shortcut	8/15/13 2:28 PM
Rpcinfo	1KB	Shortcut	8/15/13 2:28 PM
Share Editor	1KB	Shortcut	8/15/13 2:28 PM
Site Information	1KB	Shortcut	8/15/13 2:28 PM
Uninstall	1KB	Shortcut	8/15/13 2:28 PM

1 object(s) selected　391 bytes

그림 28-6. NFS 그룹

Windows 와 UNIX 의 NFS 를 리용하기전에 어느 파일체계가 유용한가 하는것을 보기로 하자.

제 15 장부터 고찰하여 온 공동탁상환경(CDE)을 사용하면 Windows 체계상에서도 UNIX 체계에서처럼 작업할수 있다.

그림 28-7 은 Terminal 창문이 열려 져 있는 공동탁상환경을 보여 주고 있다.

이 UNIX 체계우에서 처리하는 몇가지 파일체계가 있다. /home 과 /tmp 와 같은 몇가지는 아무런 제약도 받지 않고 다른것들은 제약을 받는다. 그렇지만 이 파일들을 보려고 Terminal 창문을 열어 볼 필요는 없다.

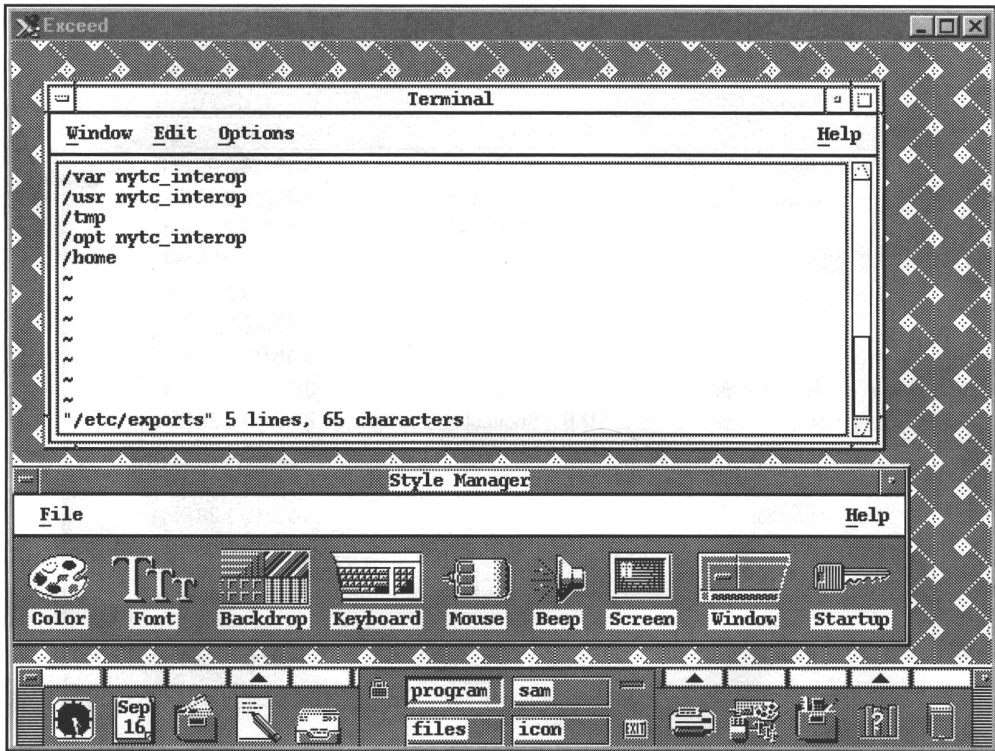



그림 28-7. /etc/export 가 보여 주는 UNIX 상우에서 공동탁상환경

Exports

Host: 15.32.165.11 Available filesystems

Filesystem	Workstations
/var (var)	nytc_interop
/usr (usr)	nytc_interop
/tmp (tmp)	(everyone)
/opt (opt)	nytc_interop
/home (home)	(everyone)

Add host to EXPORTS.HST

Exports Cancel Help

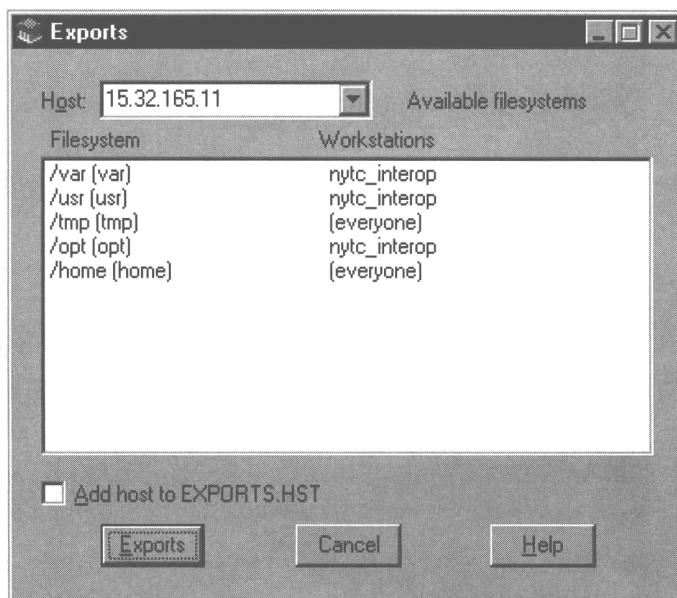



그림 28-8. 보내는 파일체계를 보여 주는 Exports 창문

그림 28-8 에 있는 창문을 열기 위해 NFS Maestro 안내를 선택할수 있다.

그림 28-8 에서 보는것처럼 IP 주소를 사용할수도 있고 또는 현재 보내는 파일체계를 보려고 하는 호스트를 규정하는 호스트이름을 리용할수도 있다. 사용자는 이 창문을 /etc/exports 파일로 만들수도 있고 기입한것들을 볼수도 있다. 현재는 그것들과 관련된 "(everyone)"와 오직 다른 체계에 설치된 체계 nytc_interop 만을 가지고 있다.

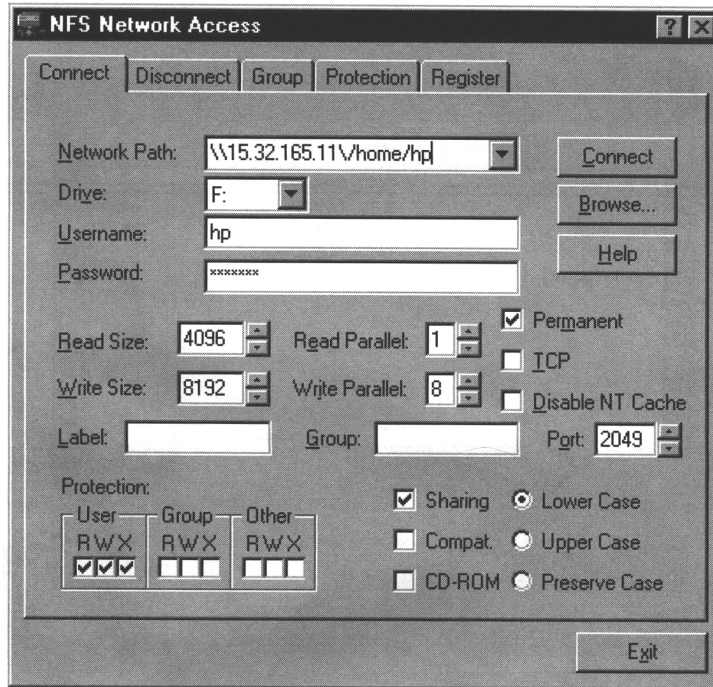



그림 28-9. F:에 /home/hp 를 태운 NFS Network Access 창문

Windows 체계우에 설치하고 싶은 UNIX 상의 파일체계들을 여러개 정할수 있다. 그림 28-9 는 Windows 체계의 구동기 F:로 설치된 UNIX 체계의 /home/hp 를 태우는 과정을 보여 준다. 여기서 주의할것은 이 파일체계가 설치될 때 UNIX 사용자이름이 hp 라는것이다.

창문의 Connect 단추를 누르면 F:에 /home/hp 가 태워 진다. NFS Maestro 를 가지고 태우기 위하여 정한 파일체계의 의미는 두가지를 담고 있는데 하나는 IP 주소 혹은 체계이름보다 높은 지위를 차지한다는것이며 다른 하나는 사용자가 IP 주소나 체계이름을 태우려는 파일체계이름이라는것이다. 주의할것은 앞의 의미가 파일체계의 이름의 일부분이라는것이다. 여기서도 체계의 IP 주소를 사용한다. Windows 체계에 태운 모든 체계를 보기 위해 Wimdows Explorer 를 기동한다.

그림 28-10 은 F:구동기에 /home/hp 를 포함하는 Explorer 창문에 태운 파일봉사체계를 보여 준다.

이 창문은 F:구동기의 /home/hp 를 보여 준다.

창문의 오른쪽 부분이 UNIX 체계상의 /home/hp 안에 있는 파일목록이다. 이 파일들은 현재 Windows 상에서 능히 호출될수 있다(이미 제시한 조종기능호출에 제시되였다.). UNIX 체계의 숨은 파일을 보기 위해 Explorer 의 Show all files 를 선택할수 있다. 사용자는 UNIX 파일들을 Windows 상의 Explorer 에서 마치 Windows 파일인것처럼 다룰수 있다. 이것은 오직 Windows 파일을 다룰수 있게 하는 계선을 뛰여 넘어 UNIX 파일로 접근할수 있게 한다.

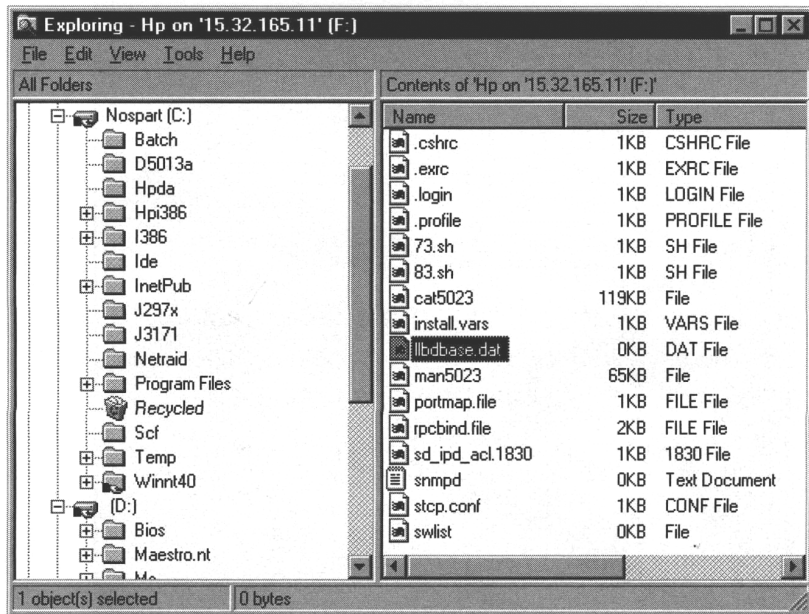



그림 28-10. F:에서 /home/hp를 보는 Windows Explorer

.cshrc Properties

General | Chmod | Chgrp

.cshrc

Type: CSHRC File
Location: F:\
Size: 814 bytes (814 bytes)

MS-DOS name: ^CSHRC.~CE
Created: Wednesday, September 16, 1998 5:37:35 PM
Modified: Wednesday, September 16, 1998 5:37:35 PM
Accessed: Wednesday, September 16, 1998 6:08:26 PM

Attributes: Read-only Hidden
Archive System

OK Cancel Apply

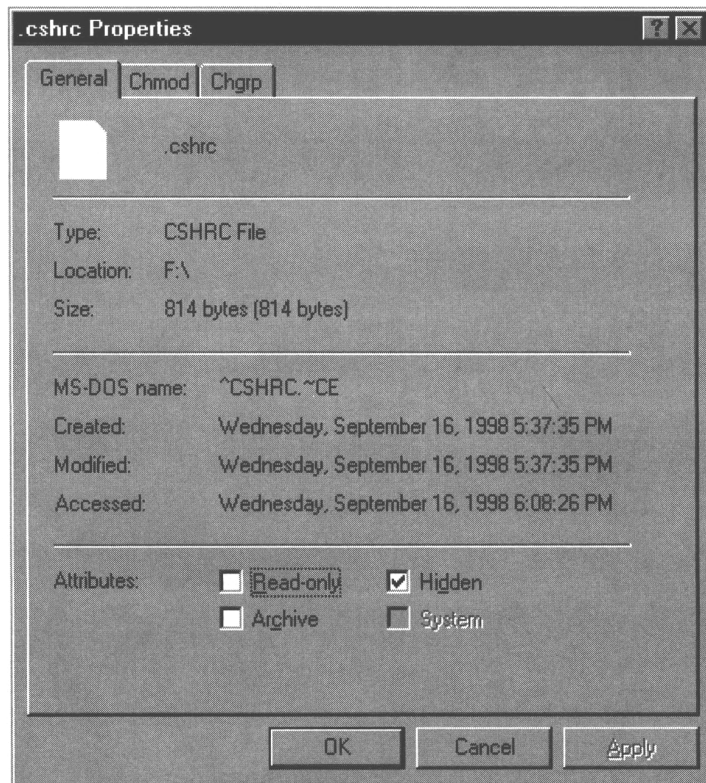



그림 28-11. .cshrc 속성의 보기

파일설치의 이러한 NFS 의 허용은 Explorer 창문에서 보이지 않는다. 그러나 개별적인 파일을 열거나 속성을 볼수 있다. 그림 28-11 은 .cshrc 파일을 보여 준다.

.cshrc 파일은 숨은 파일이며 읽기전용파일이므로 이 파일을 관리할수 있다.

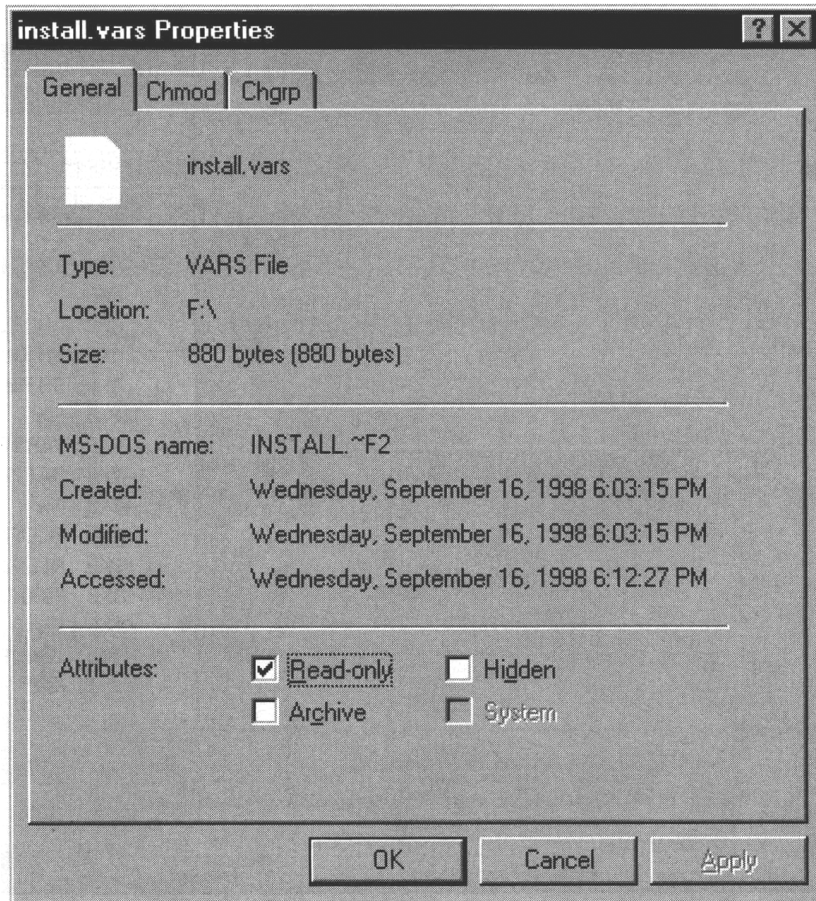



그림 28-12. install.vars 속성보기

다음으로 그림 28-12 에서 보여 준 install.vars 속성을 보자.

install.vars 는 UNIX 체계의 root 에 등록되였지만 결코 hp 라는 사용자에게 국한된것이 아니므로 읽기전용이다. hp 는 설치된 /home/hp 의 UNIX 사용자이다.

체계가 chmod 를 허용하지 않고 이 파일에서 허락을 변경시킬수 없다. 왜냐하면 이것이 root 에 종속되여 있고 그림 28-13 에서 보여 주는것처럼 /home/hp 에 hp 라는 사용자도 태웠기때문이다.

그림에 나타난 오유창문은 install.vars 의 속성이 변경되는것을 방지하기 위한것이다.

실례로 Windows NT 등록부를 UNIX 에 복사하는것을 Explorer 창문을 통하여 보여준다.

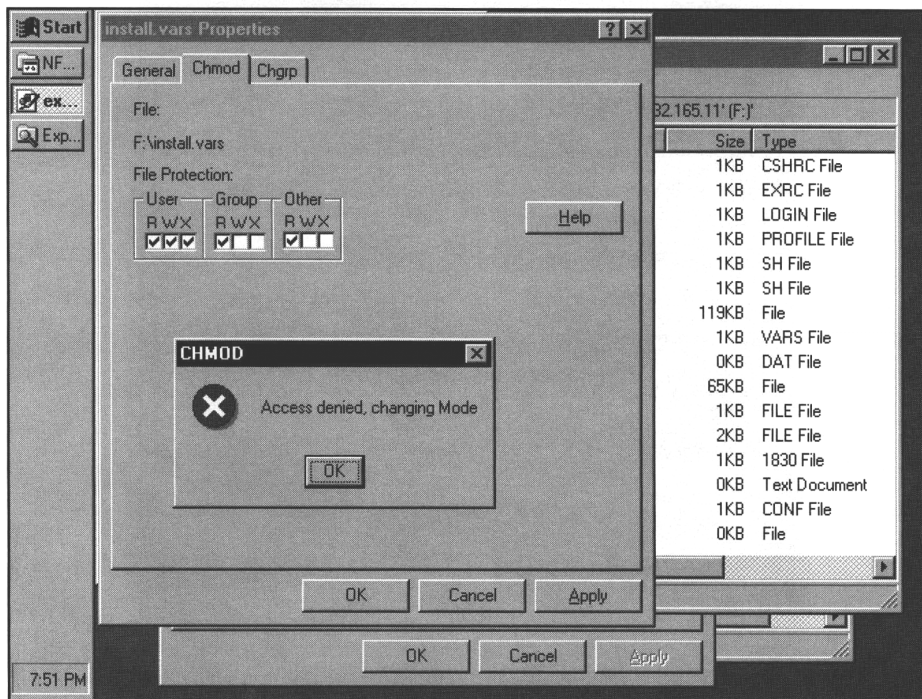



그림 28-13. install.vars 로부터 허락을 교체하기 위한 실패통보

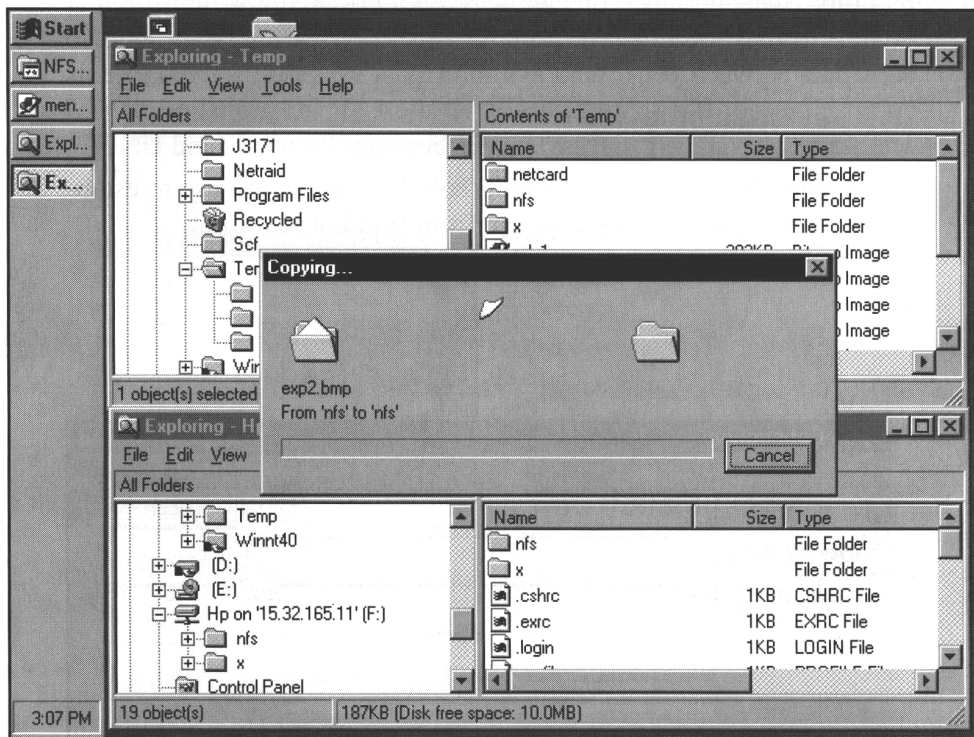



그림 28-14. Explorer 를 사용하여 Wimdows NT 등록부를 UNIX 에 복사

그림 28-14 는 Explorer 창문을 보여 준다. 꽉대기창문은 Windows 체계의 nfs 등록부를 가지고 있는데 그것은 아래창문의 UNIX 체계에 같은 이름의 등록부로 복사되고 있다. Windows 에서 UNIX 의 복사가 진행될 때(상태창이 표시된다.) 현재 복사중의 nfs 등록부안의 파일이름들이 표시된다.

Explorer 를 리용한 Windows 에서 UNIX 에로 복사는 두 조작체계사이에 파일공유가 편리하다는것을 보여 준다.

파일전송규약(FTP)

Windows 와 UNIX 상에서의 NFS 를 포괄하는 이 장은 체계호상간 조종을 위해 서술되였다. 왜냐하면 NFS 가 UNIX 파일공유에서 기본으로 되기때문이다. NFS 는 UNIX 망체계가 포함하는 자료공유를 실시간적으로 할수 있게 하는데 그우에는 Windows 체계가 앉아 UNIX 파일조작을 하게 한다. 이 방법은 파일을 공유하게 한다. 사용자는 또한 FTP 를 리용하여 Windows 와 UNIX 사이에서 파일복사를 할수 있다. NFS Maestro 의 FTP 기능이 Windows 와 UNIX 체계간의 파일전송을 간단하게 하지만 이 방법으로 결코 파일을 공유할수 없다.

그림 28-15 는 Windows 로부터 UNIX 를 접속하기 위해 리용될 대화칸을 보여 준다.

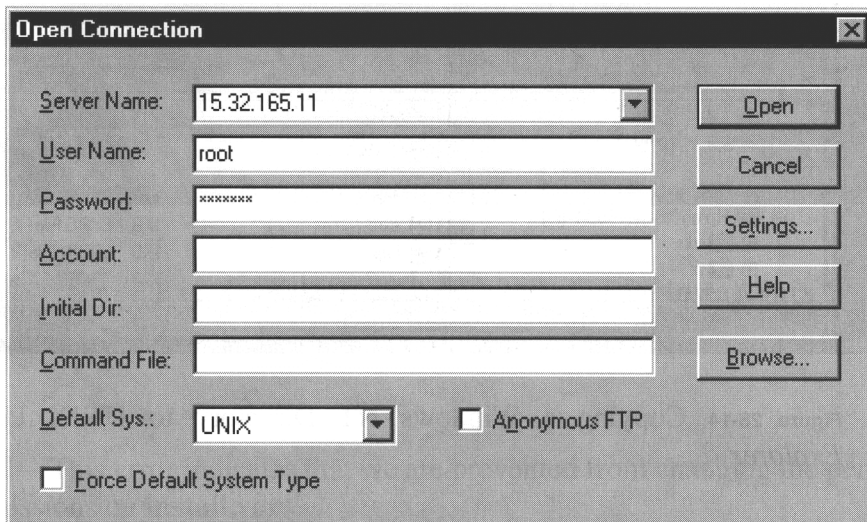



그림 28-15. Windows 로부터 UNIX 의 련결설정

이러한 접속을 설정한후 사용자가 Windows 상에서 작업하는 동안 UNIX 파일체계에서 작업할수 있는 창문이 표시된다. 그림 28-16 은 FTP 창문을 통한 UNIX 체계의 /home/hp 를 보여 준다.

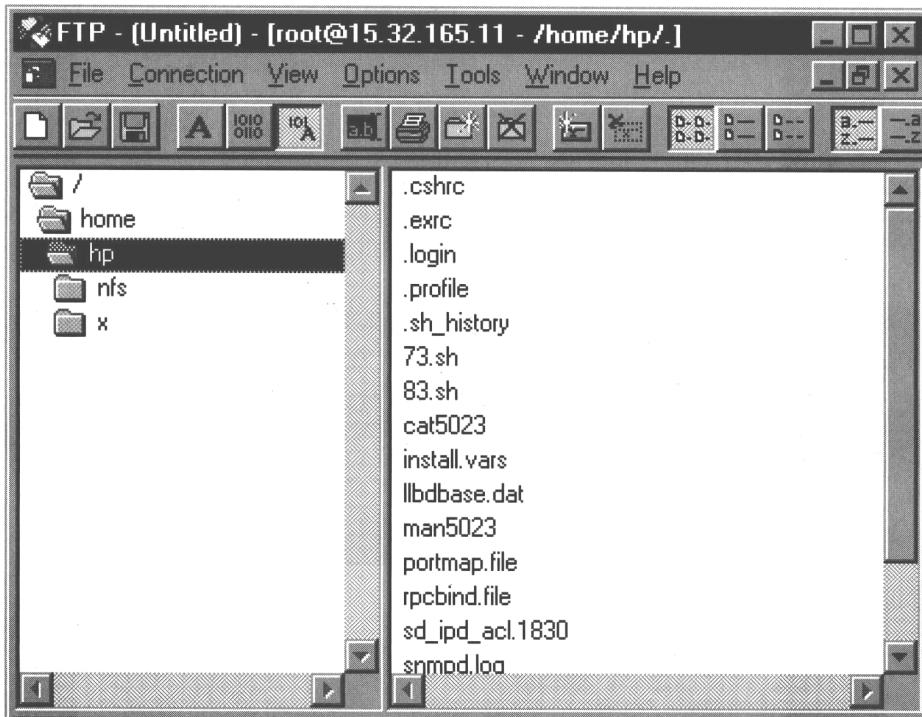



그림 28-16. FTP 창문을 사용하여 /home/hp 등록부보기

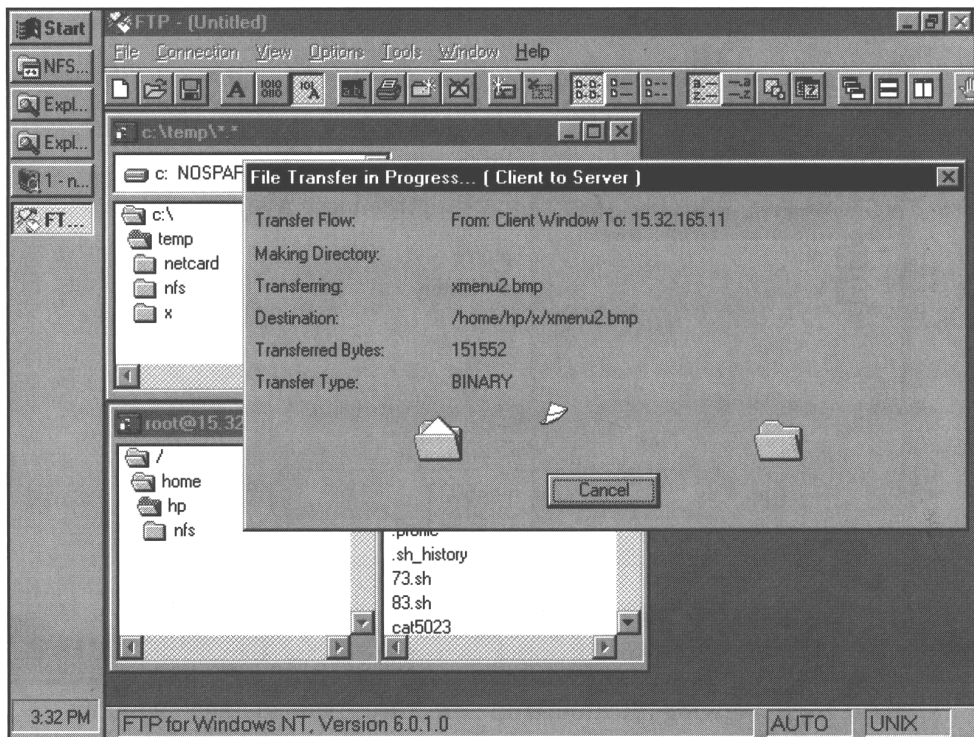



그림 28-17. Windows 로부터 UNIX 에로 등록부복사에 FTP 사용

또한 FTP 를 리용하여 파일을 복사할수 있다. 두개의 FTP 창문을 꺼내여 파일과 등록부를 한 체계에서 다른 체계에로 복사할수 있다. 그림 28-17 은 Windows 체계의 C:￥Temp￥X 를 UNIX 의 /home/hp/X 에로 복사한것이다. 이것은 두 창문의 그림기호를 사용하여 진행한다. UNIX 체계에 없던 X 등록부는 복사하여 나타난다. Windows 에서 UNIX 에로의 파일복사는 다음과 같이 한다. 우선 상태창문이 현시되고 복사중인 X 등록부(Xmenu2.bmp)파일이름이 현시된다.

FTP 가 실행될 때 여러가지 선택을 할수 있다. 그림 28-17 에서 주의할것은 전송형태가 2 진흐름이라는것이다. 이것은 전송을 초기화하기 위해 앞에서 선택한것이다.

비록 이 기능이 NFS 의 파일공유처럼 광범히 리용되지는 않지만 한 체계에서 다른 체계에로의 파일복사를 위해 널리 리용되며 더우기 Windows 와 UNIX 사이의 조종기능을 수행한다.

이 실례에서 복사될 파일을 지적하는데 그림기호를 리용한다. 또한 FTP 지령을 사용할수도 있다. 다음 내용은 지령행과 지령개요를 통하여 FTP 의 실행을 보여주는 실례이다.

파일전송규약(FTP: File Transfer Protocol) Windows 에서 UNIX 에로 전송하는것과 마찬가지로 한 체계에서 다른 체계에로 파일 혹은 다중파일을 전송한다. 아래의 실례는 체계 2(원격호스트)의 /tmp/krsort.c 를 체계 1(국부적호스트)에로 복사하는것이다.

	설 명
$ ftp system2	ftp지령표시
Connected to system2.	
system2 FTP server (Version 16.2) ready.	
Name (system2:root): root	system2에 가입
Password required for root.	
Password:	통과암호입력
User root logged in.	
Remote system type is UNIX.	
Using binary mode to transfer files.	
ftp> **cd /tmp**	system2에서 /tmp에 들어가기
CWD command successful	
ftp> **get krsort.c**	krsort.c파일입력
PORT command successful	
Opening BINARY mode data connection for **krsort.c**	
Transfer complete.	
2896 bytes received in 0.08 seconds	
ftp> **bye**	ftp벗어나기
Goodbye.	
$	

우의 실례에서 FTP 를 통하여 사용한 지령들이 조작체계에 의존되지만 두 체계가 다 UNIX 로 실행하였다. cd 지령은 등록부변경을 위해서 리용되였고 get 지령은 FTP 가 실행중인 임의의 조작체계우에서 작업할 때 사용되였다. 사용자가 FTP 지령을 어느 정도 알고 있다면 사람이라면 여러가지 망환경에서 정보전송은 그리 어렵지 않다.

FTP 가 매우 널리 리용되므로 몇가지 FTP 지령을 더 서술한다.

ftp-망을 통해 파일을 복사하는 파일전송규약

다음 목록은 몇가지 일반적으로 사용되는 FTP 지령을 서술한다.

ascii　ASCII 코드로 전송되는 파일형을 설정한다. 이것은 한 체계로부터 다른 체계로 ASCII 형 코드파일을 전송한다는것을 의미한다. 이것은 설정하지 않으면 기정적으로 된다.

실례 : ascii

binary　전송되는 파일형을 2 진으로 설정한다. 이것은 한 체계로부터 다른 체계에로 2 진파일을 전송한다는것을 의미한다. 사용자가 UNIX 가 아닌 체계에서 복사할 응용프로그람이 있는 사용자 UNIX 체계등록부를 가질것이 요구되면 사용자는 2 진전송을 사용한다.

실례 : binary

cd　원격호스트로 지적된 등록부를 교체한다.

실례 : cd/tmp

dir　원격체계등록부내용을 화면에 표시하고 국부파일이름이 지적되면 국부체계에서의 파일을 표시한다.

get　지적된 원격파일을 렬거한 국부파일에 복사한다. 사용자가 국부파일이름을 지적하지 않으면 원격파일이름을 사용한다.

lcd　지적된 등록부를 국부적호스트로 교체한다.

실례 : lcd/tmp

ls　등록부내용을 원격체계에서 화면에 표시하고 사용자가 국부파일이름을 지적하면 국부 파일체계에 대하여 표시한다.

mget　다중파일을 원격호스트로부터 국부호스트로 복사한다.

실례 : mget*.c

put　렬거된 국부파일을 렬거된 원격파일로 복사한다. 사용자가 원격파일이름을 렬거하지 않으면 국부파일이름이 사용된다.

실례 : put test.c

mput　다중파일을 국부 host 로부터 원격 host 로 복사한다.

실례 : mput*.c

system　원격호스트에서 조작체계를 실행하는 형을 보여 준다.

실례 : system

bye/quit 원격호스트접속을 닫는다.

실례:bye

다른 FTP 지령은 여기서 고찰하지 않는다.

기타 접속문제

UNIX 체계에 접속할수 있는 다른 방도도 있다. 서로 다른 체계를 접속하기 위한 두가지 기술은 FTP 와 TELNET 이다. FTP 는 이미 설명되였고 NFS Maestro 는 이 두가지를 다 장비하고 있다. TELNET 를 리용하여 Windows 상에서 UNIX 체계창문을 하나 열고 지령행을 입력할수 있다.

그림 28-18 은 호스트 Explorer 창문인데 TELNET 부분의 요소를 보여 준다.

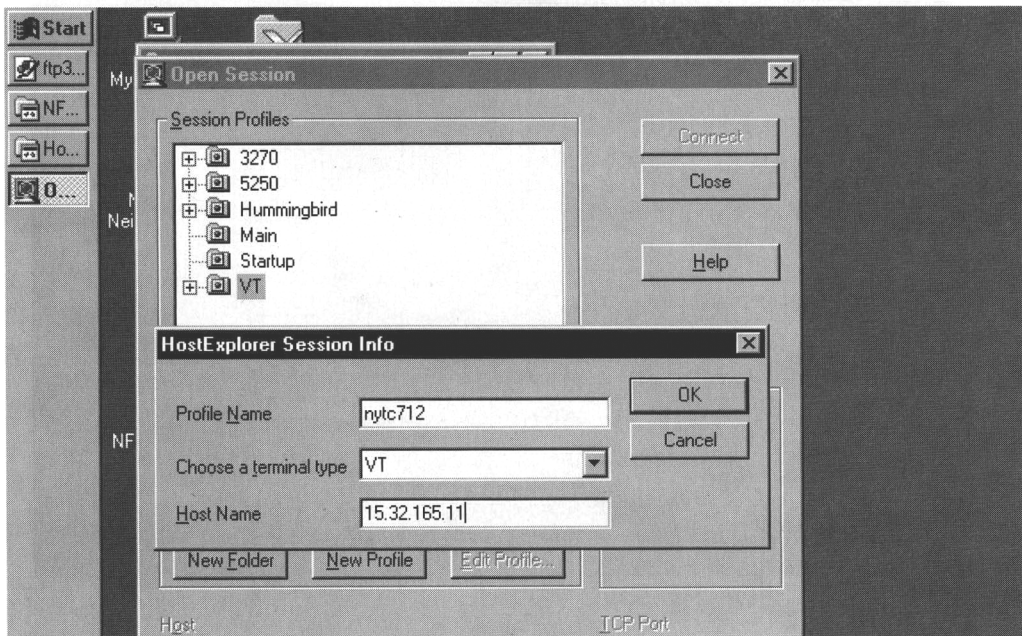



그림 28-18. Host Explorer 부분에 기호지적

호스트 Explorer 창문에서 VT 를 선택하고 그림 28-19 에서처럼 UNIX 체계에 가입할수 있다.

이 창문에서는 UNIX 체계의 종결창에 직접 마주 앉은것처럼 지령을 입구한다. 이 창문은 이미전에 복사된 nfs 등록부의 속성, 종속, 파일과 등록부그룹을 포함한 /home/hp 를 보여 준다. TELNET 는 다양한 환경에서 널리 쓰인다. X Windows 를 가지고 TELNET 기능이 아닌 도형적기능을 지원할수 있다.

```
1 - hp (15.32.165.11)
File Edit Transfer Fonts Options Macro View Window Help
$ ll -a
total 398
drwxr-xr-x   3 hp      users     1024 Sep 16 18:25 .
drwxr-xr-x   4 root    root        96 Sep 16 17:37 ..
-rw-r--r--   1 hp      users      814 Sep 16 17:37 .cshrc
-rw-r--r--   1 hp      users      347 Sep 16 17:37 .exrc
-rw-r--r--   1 hp      users      341 Sep 16 17:37 .login
-rw-r--r--   1 hp      users      446 Sep 16 17:37 .profile
-rw-------   1 hp      users       14 Sep 16 18:25 .sh_history
-rw-r--r--   1 root    sys         23 Sep 16 18:03 73.sh
-rw-r--r--   1 root    sys         23 Sep 16 18:03 83.sh
-rw-rw-rw-   1 root    sys     120922 Sep 16 18:03 cat5023
-rw-r--r--   1 root    sys        880 Sep 16 18:03 install.vars
-r--r--r--   1 root    sys          0 Sep 16 18:03 llbdbase.dat
-rw-rw-rw-   1 root    sys      66113 Sep 16 18:03 man5023
drwx------   2 hp      users     1024 Sep 16 18:17 nfs
-rw-------   1 root    sys        484 Sep 16 18:03 portmap.file
-rw-------   1 root    sys       1692 Sep 16 18:03 rpcbind.file
-rw-r--r--   1 root    sys         74 Sep 16 18:03 sd_ipd_acl.183
-rwxrwxrwx   1 root    sys          0 Sep 16 18:03 snmpd.log
-rw-r--r--   1 root    sys         60 Sep 16 18:03 stcp.conf
-rw-r--r--   1 root    sys          0 Sep 16 18:03 swlist
$
```

그림 28-19. telnet 창문

```
2 - nytc712 (15.32.165.11)
File Edit Transfer Fonts Options Macro View Window Help
    100021    4    tcp    0.0.0.0.3.111      nlockmgr    superuser
    100021    4    udp    0.0.0.0.3.112      nlockmgr    superuser
    100020    1    udp    0.0.0.0.15.205     llockmgr    superuser
    100020    1    tcp    0.0.0.0.15.205     llockmgr    superuser
    100021    2    tcp    0.0.0.0.3.113      nlockmgr    superuser
    100068    2    udp    0.0.0.0.192.11     cmsd        superuser
    100068    3    udp    0.0.0.0.192.11     cmsd        superuser
    100068    4    udp    0.0.0.0.192.11     cmsd        superuser
    100068    5    udp    0.0.0.0.192.11     cmsd        superuser
    100083    1    tcp    0.0.0.0.192.1      ttdbserver  superuser
    100005    1    udp    0.0.0.0.2.210      mountd      superuser
    100005    3    udp    0.0.0.0.2.210      mountd      superuser
    100005    1    tcp    0.0.0.0.2.211      mountd      superuser
    100005    3    tcp    0.0.0.0.2.211      mountd      superuser
    100003    2    udp    0.0.0.0.8.1        nfs         superuser
    100003    3    udp    0.0.0.0.8.1        nfs         superuser
    150001    1    udp    0.0.0.0.2.232      pcnfsd      superuser
    150001    2    udp    0.0.0.0.2.232      pcnfsd      superuser
    150001    1    tcp    0.0.0.0.2.233      pcnfsd      superuser
    150001    2    tcp    0.0.0.0.2.233      pcnfsd      superuser
1342177279    3    tcp    0.0.0.0.192.20     -           superuser
1342177279    1    tcp    0.0.0.0.192.20     -           superuser
1342177279    2    tcp    0.0.0.0.192.20     -           superuser
#
            2 Sess-1     15.32.165.11                        1 24/
```

그림 28-20. UNIX에서 rpcinfo 지령

UNIX체계우에서 실행되는 이러한 규약들은 포구들에 할당되여 있다. 그림 28-20을 통하여 이러한 포구들, 규약들, UNIX체계에서 rpcinfo지령을 사용한 련관정보를 볼수 있다.

이 창문에는 NFS와 련관하여 흥미 있는 많은 내용들이 제시되였다.

그러나 이러한 자료들을 보기 위해 rpcinfo 지령을 입구하고 UNIX 체계의 telnet 를 구성할수는 없다.

Maestro 아래의 Rpcinfo 안내에서 UNIX 호스트에 질문하고 실행을 위한 봉사를 보여준다. 그림 28-21 에서는 이 창문을 보여 준다.

Rpcinfo (Portmapper)

Host: 15.32.165.11 Registered RPC Services

Number	Ver	Port	Protocol	Program
100068	4	49163	udp	(unknown)
100068	5	49163	udp	(unknown)
100083	1	49153	tcp	(unknown)
100005	1	722	udp	mountd
100005	3	722	udp	mountd
100005	1	723	tcp	mountd
100005	3	723	tcp	mountd
100003	2	2049	udp	nfs
100003	3	2049	udp	nfs
150001	1	744	udp	pcnfsd
150001	2	744	udp	pcnfsd
150001	1	745	tcp	pcnfsd
150001	2	745	tcp	pcnfsd

RPCInfo Cancel Help

그림 28-21. Windows 에서 Rpcinfo 창문

RPC 는 원격처리호출의 략자이다. 여기에는 RPC 련관정보를 가지는 여러가지 프로그람이 있다. 몇개의 프로그람들은 Windows 와 UNIX 호상간 조종기능을 수행할것을 요구한다.

그림에서 처음에 보이는 수는 프로그람번호이다. 이것은 여러가지 프로그람에서 사용하는 RPC 번호들이다. NFS 프로그람번호는 100003 이다. 그림에서 그 다음수는 규약의 판본번호이다. 이 실례에서는 NFS 의 방안이 2 이다. 그림의 그 다음수는 포구번호이다. 포구번호는 Windows 체계의 의뢰기와 UNIX 체계에서 봉사기가 다 리용하는 포구로서 NFS 를 리용하여 전송하기 위한것이다. 그림의 다음마당의 수는 포구이다. 포구번호는 량쪽의뢰기에서 사용된다. 그것은 Windows 체계와 봉사기, UNIX 체계와 NFS 를 사용하여 통신하는데 사용된다. 그림의 다음 마당은 보통 UPP 혹은 TCP 로 사용되는 규약이다. 그림의 마지막마당은 프로그람이름이다.

NFS 의 경우에서 NFS, Portmapper, mountd, pcnfsd 는 NFS Maestro NFS 의 결과를 리용하기전에 UNIX 체계우에서 실행되고 있는가를 확인해야 한다.

Rpcinfo 는 사용자가 Windows 체계에 련결된 호스트에 대한 모든 정보를 보기 위한 일반적인 도구이다.

제２９장. UNIX 를 위한 개선된 봉사기

UNIX 상에서의 Windows 기능

지금까지는 체계호상작용기능을 고찰하기 위하여 X Windows 혹은 NFS 와 같은 UNIX 의 능률적인 기능을 Windows 에서 실현하는 방법을 취급하였다. 그러면 그 거꾸로는 못하겠는가? UNIX 에서 Windows 의 몇가지 기능을 수행할수 있다면 그것은 여러모로 리로울것이다. 망에서는 UNIX 의 인쇄기나 디스크 같은 자원들이 Windows 체계에 공유될수 있다.

이 장에는 개선된 UNIX 봉사기에 의하여 AT&T 제품으로서의 Windows 기능을 UNIX 에로 옮기기 위한 많은 결과들이 제시되였다. 개선된 UNIX 봉사기는 AT&T 와 Microsoft 회사의 합작품이다. 이것은 Windows 와 UNIX 호상간 조종을 촉진시키는 Windows 기능을 제공해 준다. 개선된 UNIX 봉사기에 의하여 UNIX 체계는 PDC 혹은 BDC, 파일봉사기, 인쇄기봉사기 혹은 어떤 Windows 기능적부분으로 구성될수 있다. 가장 권위 있는 UNIX 개발자들은 개선된 UNIX 봉사기에 의하여 제품을 만들어 낸다. 이 장에서는 개선된 봉사기/9000 이라는 UNIX 봉사기를 제공하는 HP-UX 를 고찰하기로 한다. 개선된 UNIX 봉사기의 다른 성능은 비슷한데 이 장에서는 바로 이러한 기능을 리해하기 위한데 중점을 두고 실례들을 취급한다.

이 장에서는 Windows 의 net 지령, 특히 netshare 지령을 리용한다. 이 장에서 Windows 상에서 작업할 때 지령행입구방식보다 편리한 도형식기능을 리용한다. 차이점을 리해하도록 하기 위하여 이 두가지를 함께 사용할것이다. 사용자는 이 장의 내용을 학습해감에 따라 Windows 체계의 직결도움말을 보고 싶을것이고 또한 지령행우에서의 net 지령을 설명과 함께 열람하고 싶을것이다. 아래에 몇가지 널리 쓰이는 net 지령을 요점적으로 설명하기로 한다.

net accounts	사용자계산자료기지를 유지하게 한다.
net computer	령역자료기지로부터 콤퓨터를 첨가 및 제거한다.
net config server	지령이 실행되는 봉사기의 봉사설정을 표시하거나 변경한다.
net config workstation	지령이 실행되는 워크스테이션봉사를 표시 혹은 변경한다.
net continue	net pause 지령에 의해 중지되였던 Windows 봉사를 계속하게 한다.
net files	망파일의 ID 번호표시, 공유파일에로의 접근, 파일제거 등의 조작을 위하여 사용된다.
net group	봉사기의 대역적그룹을 첨가, 표시, 변경한다.
net help	임의의 net 자령에 대한 방조를 할수 있게 한다.

net helpmsg	오유, 경고, 경보와 같은 Windows 망통보를 표시한다.
net localgroup	콤퓨터의 국부적그룹을 변경한다.
net name	콤퓨터에서 "이름통보"를 첨가, 삭제하여야 할 장소의 이름이다.
net print	과제와 공유렬의 출력목록을 위하여 사용되게 한다.
net send	망상에서 통보를 다른 사용자, 콤퓨터, "이름통보"에로 보낸다.
net session	망상에서 콤퓨터와 다른 콤퓨터사이에 목록이나 중단되는 기간을 위하여 사용되게 한다.
net share	망에 가입된 다른 콤퓨터들이 봉사기의 자료를 공유하도록 한다.
net start	봉사기와 같은 봉사를 시작한다.
net statistics	국부적워크스테이션 혹은 봉사기통계를 표시한다.
net stop	봉사기와 같은 봉사를 중지한다.
net time	령역에 있는 다른 콤퓨터의 동기화를 실현한다.
net use	공유자료를 콤퓨터에 표시, 련결 혹은 차단한다.
net user	사용자계산서를 창조 또는 변경한다.
net view	콤퓨터에 공유된 자원를 표시한다.

UNIX 상에 개선된 봉사기/9000 을 설치

사용자는 UNIX 에 개선된 봉사기/9000 을 쉽게 설치하고 구성할수 있다.

개선된 봉사기/9000 을 distributor 프로그람에 의하여 사용자의 HP-UX 체계에 설치한다. 설치가 끝나면 asu_inst 라는 구성대본을 실행해야 한다. 아래에 Windows 체계에서 hp 체계와 backup domain controll(BDC)로서 UNIX 체계 dloame 를 구성하는 asu-inst 의 실행과정을 보여 준다.

 # /opt/asu/larman/bin/asu_inst

이 지령을 주었을 때 콤퓨터화면에서의 대화는 다음과 같이 진행된다.

이 요청스크립트는 사용자에게 UNIX 체계를 위한 개선된 봉사기를 설치하고 조성하는데 필요되는 정보를 요구한다.

설치방식은 두가지이다:

고속설치(Express Setup)−설치스크립트는 기정설정값들을 리용하며 따라서 설치는 빠르고 쉽다. 설치가 완성된후에 이 설정값들을 변경시킬수 있다. 봉사기는 자기의 령역에서 기본령역조종기로 설치된다.

전용설치(Custom Setup)−이 방식은 설치를 시작할 때 설정값들을 지정할수 있게 한다. 이 방식을 선택하면 봉사기의 이름, 그것이 속하는 령역 그리고 그 령역에서의 역할을 규정해 주어야 한다.

주의: 설치는 관리급수에 따르는 통과암호를 요구한다. ‘password’가 기정통과암호로 리용되는데 설치의 마감에 다른 통과암호를 설정할수 있다.

만일 봉사기들을 설치하고 있다면 그 모든 설치들에서 기정통과암호를 리용할것을 권고한다. 망이 정확히 동작한다는것을 확정한 다음에는 그 통과암호를 변경시켜야 한다.

고속설치를 진행하겠습니까[y/n]? y

UNIX를 위한 개선된 봉사기는 여러 봉사기들의 관리를 쉽게 해주는 NETLOGON봉사를 제공한다. 단일사용자가입자료기지는 령역이라고 부르는 관리집합속에 그룹화된 여러 봉사기들에 의하여 공유될수 있다. 어느 한 령역안에서 매 봉사기는 일정한 역할을 수행한다. 기본령역봉사기라고 부르는 봉사기는 사용자가입자료기지에 대한 변경을 관리하며 그 변경들을 동일한 령역내에서 여벌령역조종기라고 부르는 다른 봉사기들에 분산시킨다. 이제 봉사기이름(망우에서 그 봉사기를 지명할)과 령역에서 그 봉사기가 수행하게 될 역할(기본봉사기 또는 여벌봉사기) 그리고 령역이름을 지정해 주어야 한다.

봉사기의 이름을 입력하든가 ‘dloanerl’를 선택하려면 Enter를 누르시오:

기본령역조종기:

관리봉사기. 사용자가입정보를 여벌령역조종기들에 분산시킨다. 망가입요청이 효력을 가지게 한다. 한개 령역에는 한개의 기본령역조종기가 있을수 있다.

여벌령역봉사기:

사용자가입정보를 기본령역조종기로부터 받는다. 망가입요청이 효력을 가지게 하며 기본령역조종기가 리용불가능하면 가입시켜 줄것을 재촉할수 있다.

역할(기본봉사기 또는 여벌봉사기): backup

이 설치는 봉사기를 여벌령역조종기로 조성한다. 기본령역조종기의 이름, 관리급수이름, 통과암호를 입력할것을 요구할수 있다. 이 설치를 성과적으로 완성시키려면 기본령역조종기는 시동되여 있어야 하며 망에 가입되여 있어야 한다.

기본령역조종기의 이름을 입력하시오: hpsystem1

봉사기 dloaner에 대한 정보:
역할 : 여벌봉사기
기본봉사기 : hpsystem1

정확합니까[y/n]? y
&a0y0C_J
기본령역조종기 hpsystem1 상에서의 관리급수이름을 입력하거나 administrator를 선택하려면 Enter를 누르시오.

이 절차는 hpsystem1 상에서의 관리급수에 대한 통과암호를 요구한다. 만일 통과암호가 설치과정에 창조된 기정통과암호(‘password’)라면 통과암호를 입력할것을 재촉할 필요가 없다. 만일 통과암호를 변경시켰다면 파일들이 설치된 후에 통과암호를 입력해야 한다.

기정통과암호를 리용하겠습니까[y/n]? y

Advanced Server/9000
Copyright（c）1988, 1991-1996 AT&T and Microsoft
Copyright（c）1992-1996 Hewlett-Packard
All rights reserved

Adding Advanced Server for UNIX Systems administrative users and groups
Add
Comment <Advanced Server account>
Home Dir </opt/asu/lanman>
UID <100>
GID <99>
Shell </sbin/false>
Name <lanman>
pw_name: lanman
pw_passwd:*
pw_uid: 100
pw_gid: 99
pw_age: ?
pw_comment:
pw_gecos: Advanced Server account
pw_dir: /opt/asu/lanman

pw_shell: /sbin/false
enter addusr
pw_name = lanman
pw_pasawd = *
pw_uid = 100
pw_gid = 99
pw_gecos = Advanced Server account
pw_dir = /opt/asu/lanman
pw_shell = /sbin/false
enter_quiet_zone（ ）
exit_quiet_zone（ ）
exiting addusr， error = 0
Add
Comment <Advanced Server Administrator>
Home Dir </var/opt/asu/lanman/lmxadmin>
GID <99>
Name <lmxadmin>
pw_name: lmxadmin
pw_passwd:*
pw_uid: 0
pw_gid: 99
pw_age:?
pw_comment:
pw_gecos: Advanced Server Administrator
pw_dir: /var/opt/asu/lanman/lmxadmin
pw_shell:
enter addusr
pw_name = 1mxadmin
pw_passwd =*
pw_uid = 0
pw_gid=99
pw_gecos = Advanced Server Administrator
pw_dir = /var/opt/asu/lanman/lmxadmin
pw_shell =
enter_quiet_zone（ ）
exit_quiet_zone（ ）
exiting addusr， error = 0
Add
Comment <Advanced Server GUEST Login>
Shell </sbin/false>

GID <99>
Name <lmxguest>
pw_name: lmxguest
pw_passwd:*
pw_uid: 0
pw_gid: 99
pw_age: ?
pw_comment:
pw_gecos: Advanced Server GUEST Login
pw_dir:
pw_shell: /sbin/false
enter addusr
pw_name = lmxguest
pw_passwd=*
pw_uid = 0
pw_gid = 99
pw_gecos = Advanced Server GUEST Login
pw_dir = /usr/lmxguest
pw_shell = /sbin/false
enter_quiet_zone（ ）
exit_quiet_zone（ ）
exiting addusr， error = 0
Add
Comment <Advanced Server World Login>
Shell </sbin/false>
GID <99>
Name <lmworld>
pw_name: lmworld
pw_passwd:*
pw_uid: 0
pw_gid: 99
pw_age: ?
pw_comment:
pw_gecos: Advanced Server World Login
pw_dir:
pw_shell: /sbin/false
enter addusr
pw_name = lmworld
pw_passwd=*
pw_uid = 0

```
pw_gid = 99
pw_gecos = Advanced Server World Login
pw_dir = /usr/lmworld
pw_shell = /sbin/false
enter_quiet_zone( )
exit_quiet_zone( )
exiting addusr, error = 0

Creating Directory: /home/lanman
Setting owner, group, and permissions for installed files . . . .

Enter the password for administrator on hpsysteml:
Re-enter password:

Contacting the server 'hpsysteml' . . . Success

Creating Advanced Server for UNIX Systems accounts database.

Starting the Advanced Server for UNIX Systems . . .

The Advanced Server for UNIX Systems is now operational.
#
```

설치구성이 끝난후 net demon 을 실행하는것은 아래에서 ps 지령입력결과가 보여 주는것처럼 개선된 봉사기/9000 의 필수적구성부분으로 된다.

```
#ps -ef |grep netdemon
    root 1100 1 0 10:18:38 ?    0:00 /opt/lmu/netbios/bin/netdemon
#
```

net demon 의 첨가에서는 METBIOS 가 반드시 기동한다. Net demon 첨가에서는 UNIX 상의 여러개 처리가 개선된 봉사기/9000 에 의하여 시작된다. 또한 ps 지령을 통해 그 처리를 보면 개선된 봉사기/9000 이 실행하고 있다는것을 알수 있다.

```
# ps -ef | grep lm

root    3285     1   0   10:37:19   ?   0:00   lmx.dmn
root    3200     1   0   10:36:57   ?   0:00   lmx.ctrl
root    3262  3200   0   10:37:07   ?   0:00   lmx.Srv –s 1
```

```
root        3295        1    0    10:37:20    ?    0:00    1MX.Bched
root        3289        1    0    10:37:19    ?    0:00    lmx.browser
root        1100        1    0    10:18:38    ?    0:00    /opt/lmu/netbios/bin/netdemon
#
```

많은 처리가 제시되여 있는데 그중에서 프로그람 lmx.dmn, 조종처리 lmx．ctrl, 작성자 lmx．sched, 열람기 lmx．brower, 의뢰기부분인 lmx．ser 들을 볼수 있다. 개선된 봉사기/9000 이 기동하지 않으면 봉사기기동을 위해 nst start server 지령을 실행한다. 봉사를 끝내기 위해 net stop server 지령을 사용한다.

보충적으로 Windows 상에서 간단히 개선된 봉사기/9000 의 사용을 할수 있도록 UNIX 상에 작성된 여러가지 봉사자들과 그룹이 주어 져 있다. /etc/passwd 에 새로운 사용자들이 있고 /etc/group 에는 새로운 그룹들이 있다.

```
# cat /etc/passwd
root:jThTuY90hNxGY:0:3::/:/sbin/sh
daemon:*:1:5::/:/sbin/sh
bin:*:2:2::/usr/bin:/shin/sh
sys:*:3:3::/:
adm:*:4:4::/var/adm:/sbin/sh
uucp:*:5:3::/var/spool/uucppublic:/usr/lbin/uucp/uucico
lp:*:9:7::/var/spool/lp:/sbin/sh
nuucp:*:11:11::/var/spool/uucppublic:/usr/lbin/uucp/uucico
hpdb:*: :1:ALLBASE:/:/sbin/sh
nobody:*:-2:-2147483648::/:
lanman:*:100:99:Advanced Server account:/opt/asu/lanman:/sbin/false
lmxadmin:*:202:99:Advanced Server Administrator:/var/opt/asu/lanman/lmxadmin:
lmxguest:*:203:99:Advanced Server GUEST Login:/usr/lmxguest:/sbin/false
lmworld:*:204:99:Advanced Server World Login:/usr/lmworld:/sbin/false
# cat /etc/group
root::0:root
other::l:root, hpdb
bin::2:root, bin
sys::3:root, uucp
adm::4:root, adm
daemon::5:root, daemon
mail::6:root
lp::7:root, lp
tty::10:
nuucp::11:nuucp
```

users::20:root
nogroup:*:-2:
DOS---- ::99:lanman
DOS-a--::98:lanman
DOS--s-::97:lanman
DOS --- h::96:lanman
DOS-as-::95:lanman
DOS-a-h::94:lanman
DOS--sh::93:lanman
DOS-ash::92:lanman
#

자동적으로 진행되는 UNIX 체계변경에 보충할것은 windows PDC 가 UNIX 체계를 열때 BPC 로 리해하여야 한다는것이다.

그림 29-1 은 hpsystem1 에서 산생되는 화면인데 PDC 이다. 화면은 BDC 로 작용하는 dloaner 의 UNIX dloaner 의 기정적공유등록부를 보여 준다. C:\opt\asu\lanman 와 같은 공유속성도 볼수 있다.

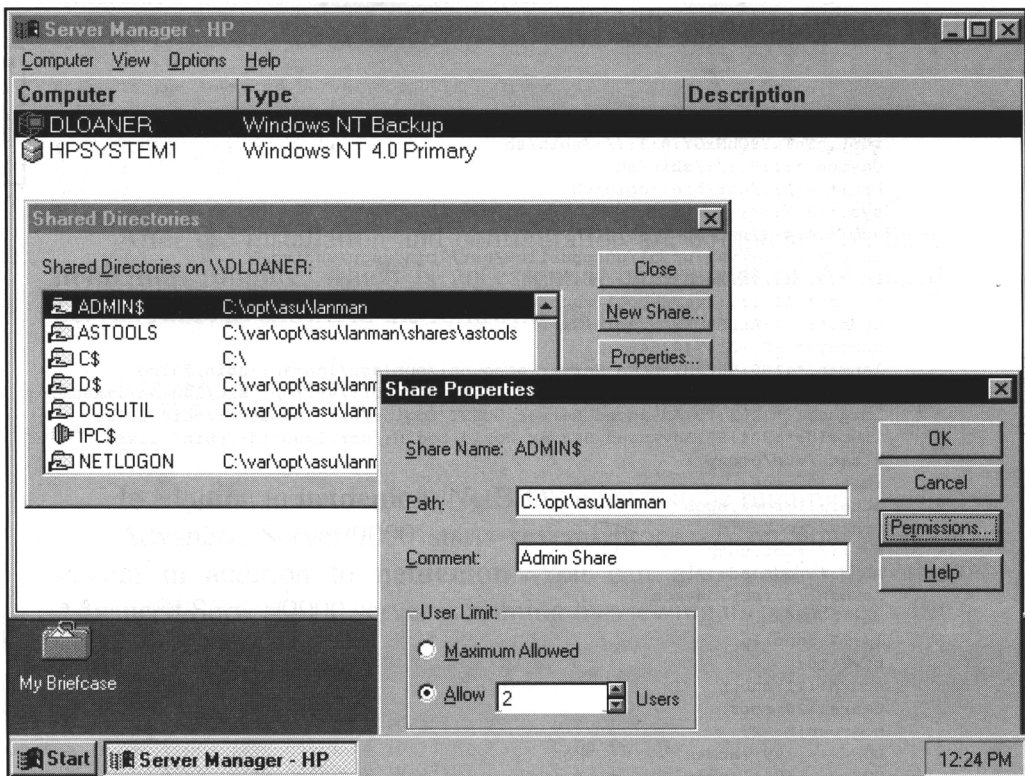



그림 29-1. 개선된 봉사기/9000 을 적재하고
공유한 다음의 기정적공유등록부

아래와 같이 UNIX 상의 net 지령을 사용하여 볼수도 있다.

```
# /opt/asu/lanman/bin/net share
Sharename     Resource                              Remark
-------------------------------------------------------------------------------------
ADMIN$        C:\OPT\ASU\LANMAN                     Admin Share
IPC$                                                IPC Share
C$            C:\                                   Root Share
D$            C:\VAR\OPT\ASU\LANMAN\SHARES          SystemRoot Share
ASTOOLS       C:\VAR\OPT\ASU\LANMAN\SHARES...       Advanced Server Tools
DOSUTIL       C:\VAR\OPT\ASU\LANMAN\SHARES...       DOS Utilities
NETLOGON      C:\VAR\OPT\ASU\LANMAN\SHARES...       Logon Scripts Directory
PATCHES       C:\VAR\OPT\ASU\LANMAN\SHARES...       Client Patches
PRINTLOG      C:\VAR\OPT\ASU\LANMAN\SHARES...       LP printer messages
USERS         C:\HOME\LANMAN                        Users Directory
The command completed successfully.
#
```

이 실례에는 개선된 봉사기/9000 에 의해 설치된 기정적인 공유등록부들이 있다. 이것들은 기호 $에 잇달리는데 조작계획을 위해서 숨겨 진 공유등록부들이다. Windows Explorer 를 기동하면 이 숨은 등록부를 볼수 없다. 앞에서 본 인쇄기와 디스크 등을 보충적으로 공유할수 있는데 "인쇄기공유"와 "파일체계공유" 등을 들수 있다.

인쇄기공유

개선된 봉사기/9000 에 의한 기정공유에는 Windows 와 UNIX 사이에서 보충적으로 공유하려고 하는 자원들이 있다. 실례로 UNIX 상의 인쇄기를 Windows 에서 호출할수 있다. 아래의 지령들에서 UNIX 에서 인쇄기공유를 첨가하는것을 볼수 있다. 첫 지령은 UNIX 의 lpstat 인데 인쇄기 laser 의 존재상태를 나타낸다.

```
# lpstat -t
scheduler is running
system default destination: laser
device for laser: /dev/c2t0d0_lp
laser accepting requests since Feb 11 17:23
printer laser is idle. enabled since Feb 11 17:23
fence priority : 0
no entries
#
```

다음의 지령은 인쇄기 laser 를 공유인쇄장치로서 지적하고 net 지령으로 실행하는것이다.

```
# /opt /asu /lanman /bin /net net share laser=laser /print
laser was successfully shared
```

인쇄기의 구성에 대하여 보기 위해서는 net print 지령을 사용하여야 한다.

```
# net print laser /options
Printing options for LASER

Status                    Queue Active
Remark
Print Devices             laser
Driver                    HP-UX LM/X Print Manager
Separator file
Priority                  5
Print after               12:00 AM
Print until               12:00 AM
Print processor
Parameters                COPIES=1 EJECT=AUTO BANNER=YES
The command completed successfully.
#
```

개선된 봉사기/9000 이 실행중인 UNIX 체계에 련결한후에는 인쇄기에 대한 알기 쉬운 정보를 볼수 있다. 개선된 봉사기/9000 의 인쇄기는 raw 구성이 아니다. 아래에서는 인쇄기를 raw 로 구성하는데 대하여 보여 주고 있다.

```
# net print laser /parms:types=-oraw
The command completed successfully.
```

TYPES=-oraw 에 의한 새로운 구성은 다음의 출구를 통해 볼수 있다. 이 장치는 laser 가 련결된 개선된 봉사기/9000 이 실행된 UNIX 체계에 대하여 Windows 에서 인쇄된다.

```
# net print laser /options
Printing options for LASER

Status                    Queue Active
Remark
Print Devices             laser
Driver                    HP-UX LM/X Print Manager
Separator file
Priority                  5
Print after               12:00 AM
Print until               12:00 AM
Print processor
Parameters                COPIES=1TYPES=-orawEJECT=AUTO BANNER=YES
The command completed successfully.
#
```

net 지령에 의하여 공유된 모든 장치를 볼수 있다.

```
# /opt/asu/lanman/bin/net share
Sharename     Resource                                Remark
-------------------------------------------------------------------------------
ADMIN$        C:\OPT\ASU\LANMAN                       Admin Share
IPC$                                                  IPC Share
C$            C:\                                     Root Share
D$            C:\VAR\OPT\ASU\LANMAN\SHARES            SystemRoot Share
ASTOOLS       C:\VAR\OPT\ASU\LANMAN\SHARES…           Advanced Server Tools
DOSUTIL       C:\VAR\OPT\ASU\LANMAN\SHMES…            DOS Utilities
NETLOGON      C:\VAR\OPT\ASU\LANMAN\SHARES…           Logon Scripts Directory
PATCHES       C:\VAR\OPT\ASU\LANMAN\SHARES…           Client Patches
PRINTLOG      C:\VAR\OPT\ASU\LANMAN\SHARES…           LP printer messages
USERS         C:\HOME\LANMAN                          Users Directory
LASER         laser                                   Spooled
The command completed successfully.
#
```

마지막항목에서는 net 지령에 의해 보충된 인쇄기 laser 에 대하여 보여 준다. 개선된 봉사기/9000 에서 실행하는 UNIX 체계의 모든 지령을 사용하였다. 인쇄기 laser 가 공유장치라는것을 강조하기 위하여 explorer 를 사용하면 그림 29-2 에서 보여 주는것처럼 dloaner 공유장치를 Windows 에서 볼수 있다.

Control Panel 의 Printers 에서 공유인쇄기에 대한 설명을 볼수 있다.

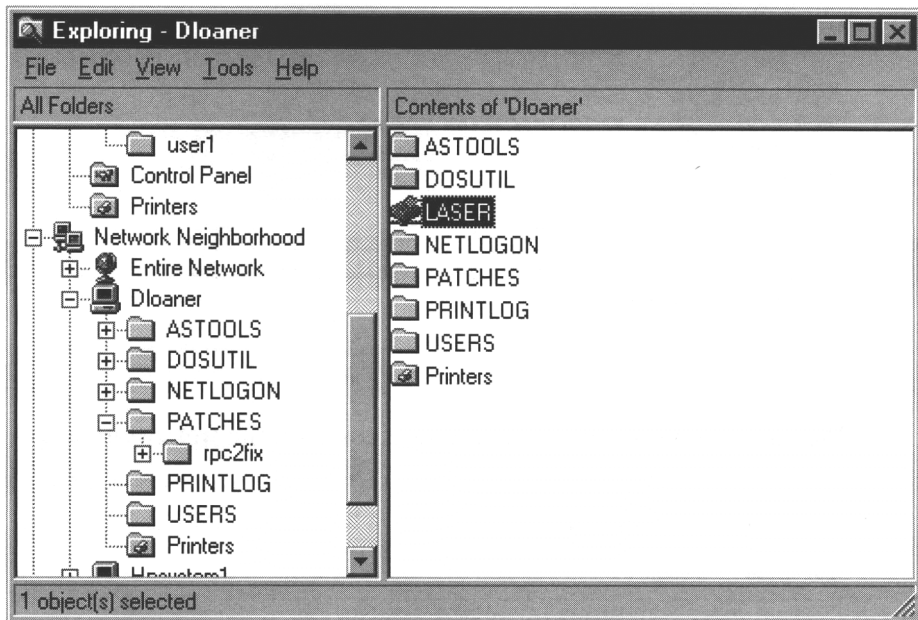



그림 29-2. 인쇄기 Laser 를 보는 Windows Explorer

파일체계공유

개선된 봉사기 /9000 을 실행하는 UNIX 체계에 인쇄기를 첨가하여 공유하는것을 다음의 실례에서 보여 준다.

```
# /opt/asu/lanman/bin/net  share
Sharename      Resource                                  Remark
------------------------------------------------------------------------------------------------
ADMIN$         C:\OPT\ASU\LANMAN                         Admin  Share
IPC$                                                     IPC Share
C$             C:\                                       Root Share
D$             C:\VAR\OPT\ASU\LANMAN\SHARES              SystemRoot Share
ASTOOLS        C:\VAR\OPT\ASU\LANMAN\SHARES…             Advanced Server Tools
DOSUTIL        C:\VAR\OPT\ASU\LANMAN\SRARES…             DOS Utilities
NETLOGON       C:\VAR\OPT\ASU\LANMAN\SHARES…             Logon Scripts Directory
PATCHES        C:\VAR\OPT\ASU\LANMAN\SHARES…             Client Patches
PRINTLOG       C:\VAR\OPT\ASU\LANMAN\SHARES…             LP printer messages
USERS          C:\HOME\LANMAN                            Users Directory
LASER          laser                                     Spooled
The  command  completed  successfully.
#
```

이것들은 보충인쇄기들을 포함하고 있다. Net share 지령을 리용하여 UNIX 체계를 공유한다. UNIX 체계 dloaner 우의 /nome 등록부를 공유하기 위해서는 다음지령을 사용하면 된다.

```
# /opt /asu /lanman /bin /net  share  home=c: /home
home  was  shared  successfully.
```

주의할것은 Windows 에서 사용한 등록부구분기호 "\"와 달리 UNIX 는 “/”를 리용한다는것이다. net 지령에 의하여 새 HOME 를 가진 dloaner 우의 공유에 대하여 볼수 있다.

```
# /opt/aou/lanman/bin/net  share
Sharename      Resource                                  Remark
------------------------------------------------------------------------------------------------
ADMIN$         C:\OPT\ASU\LANMAN                         Admin Share
IPC$                                                     IPC Share
C$             C:\                                       Root Share
D$             C:\VAR\OPT\ASU\LANMAN\SHARES              SyBtemRoot Share
ASTOOLS        C:\VAR\OPT\ASU\LANMAN\SHARES…             Advanced Server Tools
DOSUTIL        C:\VAR\OPT\ASU\LANMAN\SHARES…             DOS Utilities
HOME           C:\HOME
```

```
NETLOGON     C:\VAR\OPT\ASU\LANMAN\SHARES···     Logon Scripts Directory
PATCHES      C:\VAR\OPT\ASU\LANMAN\SHARES···     Client Patches
PRINTLOG     C:\VAR\OPT\ASU\LANMAN\SHARES···     LP printer messages
USERS        C:\HOME\LANMAN                      Users Directory
LASER        laser                               Spooled
The command completed successfully.
#
```

이것을 그림 29-3 에서처럼 Windows 에서도 볼수 있다.

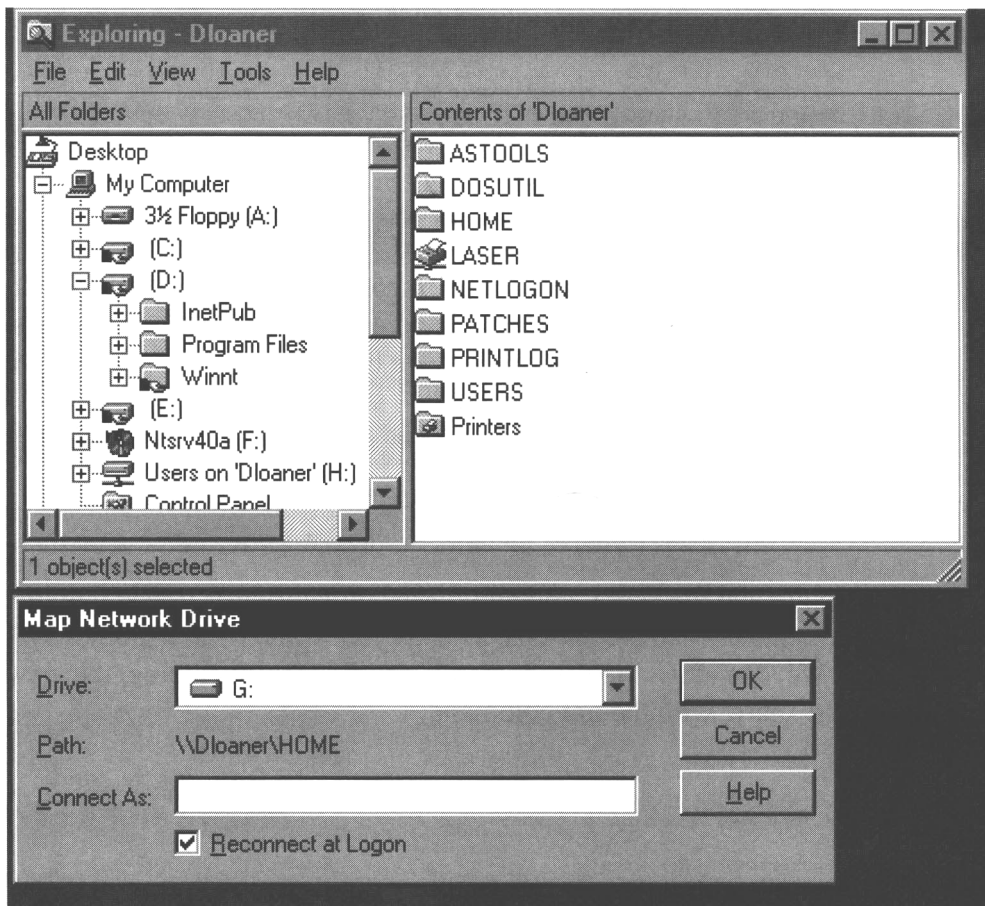



그림 29-3. 새로운 공유 HOME 을 보는 Windows Explorer

이 장에서는 개선된 봉사기/9000 의 부분적인 기능만을 취급하였다. Windows 에 의해 UNIX 련결인쇄기와 디스크를 공유하는 UNIX 체계에 대하여 간단히 언급해 두자. 이것도 잘 알려 진 개선된 봉사기/9000 의 기능이다. Windows 에서 하던 조작과 류사하게 개선된 봉사기/9000 에서 진행할수 있으므로 이 장에서 취급한 내용이 전부가 아니라는것을 강조해 준다.

제３０장. Windows 지령행 : NET 지령, POSIX 편의프로그람

UNIX 체계관리자에 대한 개괄

UNIX 체계관리는 대부분 지령행에 의하여 실행된다. 많은 체계관리의 루틴과제들이 실행될수 있는 대단히 좋은 체계관리대면부들도 개발되여 있다. 그렇지만 아직도 대부분 UNIX 체계관리자들은 지령행을 어떤 그림도구보다도 더 많이 사용하고 있다. 이러한것은 Windows 에서도 마찬가지이다. Windows 에서는 대부분 체계관리함수들을 《지적하고 눌러》서 실행하군 한다. 그러나 Windows 안에는 많은 함수들을 지령행으로 실행할수 있다. UNIX 와 Windows 호상조작성을 취급한 장에서 Windows 지령행은 Windows 에서 UNIX 기능을 구체적으로 주는 일반적인 POSIX 봉사프로그람그룹을 가지고 있었다.

Windows 의 지령은 "NET"에 의하여 시작하는데 이 지령은 Windows 를 위한 체계관리지령이다. 여기서는 또한 Windows 에서 grep, ls 등과 같은 UNIX 함수를 Windows 지령행에서 실행할수 있는 POSIX 지령을 취급할것이다. 이 장의 마지막에서는 여벌복사와 같은 과제의 수행과 지령을 실행하는 프로쎄스를 보여 주는 일부 보충적인 Windows 지령을 고찰하기로 한다.

POSIX 지령은 실제적으로 UNIX 와 Windows 의 부분처럼 구성되여 있다. 이밖의 Windows 지령들은 UNIX 체계관리자들이 리용할수 있는 Windows 체계관리지령들의 형태들에 대한 기본방법을 제시하여 준다.

Windows 지령행

모든 Windows 조작체계는 도형방식사용자대면부를 통하여 체계관리과제를 실현하는데 기초를 두고 있다.

이 장의 목적은 지령행에서 나오는 유용한 지령들을 보여 주는것이다. 사용자는 지령과 같은것을 사용하는것이 사용자 Windows 봉사기의 관리를 위해 쓸모가 있는가, 없는가를 선택하며 또한 Windows 도형사용자대면부만을 사용할것을 요구하는가를 확증할수 있다.

NET 지령

여기서는 Windows 의 net 지령의 일부를 서술하였다. 아래에서는 망지령들을 폭 넓게 리용하기 위한 목록과 간단한 설명을 하고 있다.

net accounts	사용자의 계산자료기지를 유지하는데 사용한다.
net computer	령역자료기지로부터 콤퓨터를 첨가 혹은 삭제하는데 사용된다.
net config server	지령이 실행되는 봉사기의 봉사에 대한 설정을 변경 또는 표시한다.
net config workstation	지령이 실행되는 워크스테이션봉사에 대한 설정을 표시 또는 변경한다.
net continue net pause	지령에 의하여 정지된 Windows 봉사를 계속한다.
net files	ID 번호를 기록하는것, 공유된 파일을 닫는것, 열쇠걸린 파일을 삭제하는것 등과 같은 망파일조작을 위하여 사용한다.
net group	봉사기에서 첨가, 표시, 또는 대역그룹의 변경을 위하여 사용된다.
net help	임의의 망지령에 대한 방조선택의 목록을 표시한다.
net helpmsg	오유, 경고, 경보통신과 같은 Windows 망통지문의 설명을 표시한다.
net localgroup	콤퓨터상에서 국부그룹을 변경하는데 사용한다.
net name	콤퓨터에서 "이름통보문"을 첨가, 삭제하는데 이 이름은 통보문을 보내기 위한 이름이다.
net print	목록인쇄일감과 공유렬들을 표시하기 위하여 사용된다.
net send	망상에서 다른 사용자들, 콤퓨터, "이름통보문"에 대한 대답통보를 보낸다.
net sesstion	망상에서 콤퓨터와 다른 콤퓨터사이의 련결 및 차단기간을 설정한다.
net share	망상에서 다른 콤퓨터들과 봉사기자원을 함께 공유한다.

net start	봉사기가 수행하는것과 같은 봉사를 시작한다.
net statistics	봉사기봉사 또는 국부워크스테이션의 통계일지를 표시한다.
net stop	봉사기에서 수행하는것과 같은 봉사를 정지한다.
net time	령역에서 다른 콤퓨터에 의하여 콤퓨터시간을 맞춘다.
net use	공유된 자원을 콤퓨터에 표시, 련결, 차단을 한다.
net user	사용자계산서를 창조하거나 수정한다.
net view	공유되여 있는 자원을 콤퓨터상에서 보여 준다.

아래에서는 우의 지령들을 리용한 실례들과 일부 망지령에 대하여 간단한 설명을 하기로 한다.

많은 문장들이 지령이름뒤에 /?를 써서 지령개요를 포함한다. 실례로 NET ACCOUNTS지령을 위한 지령개요를 제시하기 위하여 사용자는 다음과 같이 쓸수 있다.

C:**NET ACCOUNTS**/?

다음 실례에서 보여 준것처럼 사용자는 역시 HELP 지령이름을 사용하면 상세한 도움말정보를 얻을수 있다.

C:**HELP NET ACCOUNTS**

NET ACCOUNTS

NET ACCOUNTS-사용자계산자료기지를 보존한다.

NET ACCOUNTS지령형식 :

C:**NET ACCOUNTS**/?
NET ACCOUNTS [/FORCELOGOFF:{minutes | NO}] [/MINPWLEN:length]
[MAXWAGE:{days | UNLIMITED}] [/MINPWAGE:days]
[/UNIQUEPW: number] [/DOMAIN]
NET ACCOUNTS [/SYNC]

일반적으로 사용되는 선택항목들은 다음과 같다.

/DOMAIN　　현재 콤퓨터보다 오히려 령역조종자우에서 기록된 동작이 실행된다.

/FORCELOGOFF

출력행에 수자들이 표시된다.

/MINPWLEN:length

length 에 암호문자의 최소개수가 기록된다.

/MAXPWAGE:days

암호가 효과을 가질수 있는 날자의 웃한계를 기입하거나 암호를 제한없이 사용하기 위한 인수를 기입한다.

/MINPWAGE:days

암호를 교체하기전에 통과하는 날자의 아래한계를 기록한다.

/UNIQUEPW:number

매 사용자암호는 number 에 기록된 수로 유일하게 교체된다.

/SYNCH　　계산자료기지가 동시에 만들어 지게 한다.

다음의 실례는 선택항목을 사용하지 않는 NET ACCOUNT 를 보여 준다.

C:\ **NET ACCOUNTS**

Force user logoff how long after time expires?:	Never
Minimum password age（days）:	0
Maximum password age（days）:	42
Minimum password length:	0
Length of password history maintained:	None
Lockout threshold:	Never
Lockout duration（minutes）:	30
Lockout observation window（minutes）:	30
Computer role:	BACKUP
Primary Domain controller for workstation domain:	\\NISDEV

The command completed successfully.

아래에서는 암호의 최소길이가 5 개 문자로 교체되도록 하는 선택항목 MINPWLEN 을 리용한 NET ACCOUNT 를 보여 준다.

C:**NET ACCOUNTS/MINPWLEN:5**

The request will be processed at the primary doamin controller for domain
NSDNIS

The command completed successfully.

선택항목이 없이 NET ACCOUNT 를 리용하면 새로운 가장 작은 암호길이로서 처리된다.

C:\ **NET ACCOUNTS**

Force user logoff how long after time expires?:	Never
Minimum password age (days):	0
Maximum password age (days):	42
Minimum password length:	5
Length of password history maintained:	None
Lockout threshold:	Never
Lockout duration (minutes):	30
Lockout observation window (minutes):	30
Computer role:	BACKUP
Primary Domain controller for workstation domain:	\\NISDEV

The command completed successfully.

NET COMPUTER

NET COMPUTER-령역자료기지로부터 콤퓨터들을 첨가하거나 삭제한다.

NET COMPUTER지령형식 :

C:**NET COMPUTER**/?
NET COMPUTER \\computername {/ADD | /DEL}

일반적으로 사용되는 선택항목들은 다음과 같다.

computername	첨가되였거나 삭제된 콤퓨터이름
/ADD	콤퓨터첨가
/DEL	콤퓨터삭제

다음 실례는 콤퓨터를 첨가하는 /ADD선택항목에 대하여 보여 준다.

C:\ NET COMPUTER\\ SYSTEM2 /ADD

The request will be processed at the primary domain controller for domain NSDNIS.

The command completed successfully.

NET CONFIG SERVER

NET CONFIG SERVER-관리자들의 그룹성원처럼 사용자는 봉사설정을 변경할수 있다.

NET CONFIG SERVER 지령형식 :

C:**NET CONFIG SERVER**/?

NET CONFIG SERVER [/AUTODISCONNECT:time]
[/SRVCOMMENT:"text"]
[/HIDDEN:{YES｜NO}]

일반적으로 사용되는 선택항목들은 다음과 같다.

/AUTODISCONNECT:time　　time 은 작용하는 계수가 끝나기전에 통과하는 분을 지적하기 위하여 사용한다.

/SRVCOMMENT:"text"

/HIDDEN:｛YES｜NO｝

NET CONTINUE

NET CONTINUE-NET PAUSE 에 의해 중지된 Windows 봉사를 복귀시킨다.

NET CONTINUE 지령형식 :

C:**NET CONTINUE**/?

NET CONTINUE service

사용자는 NET PAUSE 및 NET CONTINUE 지령을 리용하여 많은 Windows 봉사를 멈출수도 있고 계속할수 있다.

아래에서는 NET LOGON 봉사를 멈추기 위하여 NET PAUSE 지령을 사용하는것과 그때에 NET CONTINUE 지령으로 재설정하는데 대하여 보여 준다.

그림 30-1 은 체계의 봉사대화칸인데 이것은 체계상에서 실행중인 봉사를 보여 준다.

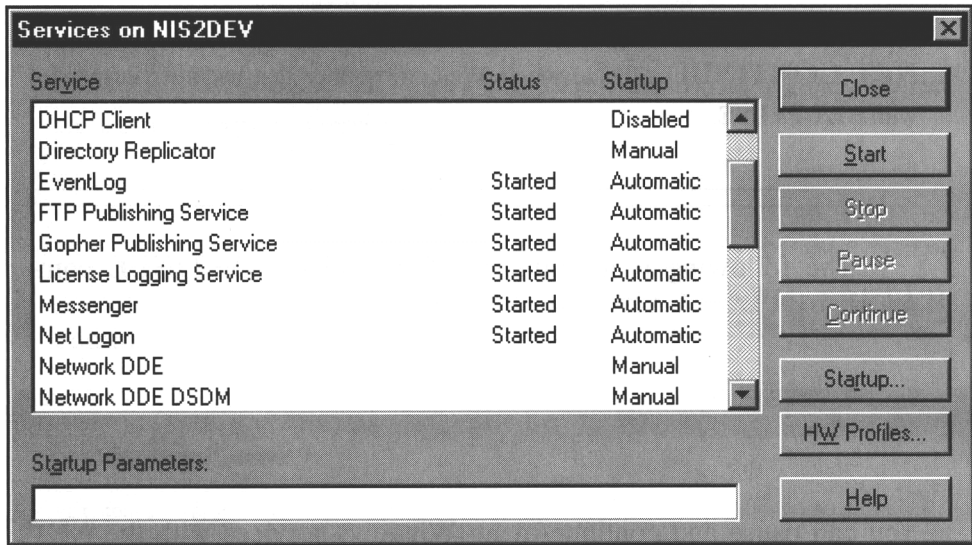



그림 30-1. Net Logon 이 시작된 대화칸의 봉사

다음은 NET LOGON 봉사를 중지시키기 위하여 NET PAUSE 지령을 사용한다.

C:**NET PAUSE NTLOGON**
The Net Logon service was paused successfully.

그림 30-2 는 대화칸의 봉사과정을 다시 보여 주는데 NET LOGON 봉사가 중지된것을 보여 준다.

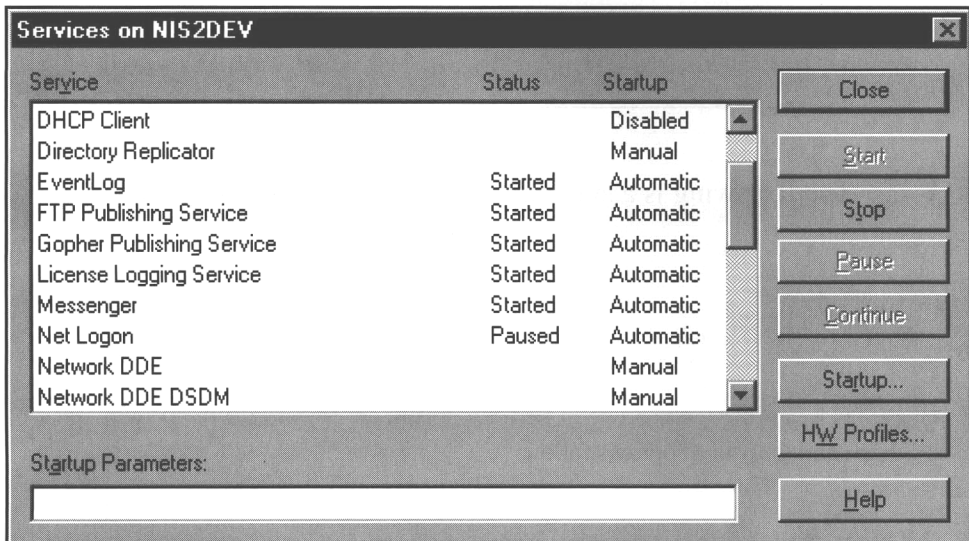



그림 30-2. Net Logon 이 중지된 대화칸봉사

다음의 NET CONTINUE 지령으로 NET LOGON 봉사를 회복할수 있다.

C:**NET CONTINUE NTLOGON**
The Net Logon service was continued successfully.

NET FILE

NET FILE-이 지령은 파일의 표시와 열기, 닫기를 한다.

NET FILE 지령형식 :

C:**NET FILE**/?

NET FILE [id[/CLOSE]]

일반적으로 사용되는 선택항목들은 다음과 같다.

id	사용자들이 식별하기 위한 번호를 지적한다.
id/CLOSE	id 번호에 의하여 지적된 파일을 닫는다.

다음의 실례들은 NET FILE 지령을 선택항목이 없이 표시하는것에 대하여 보여 준다.

C:\ NET FILE

ID	Path	User name	# Locks
97	C:\tif_map_proj\sql	leung_k	0
147	\PIPE\Bamr	administrator	0
148	\PIPE\lsarpc	administrator	0

The command completed successfully.

이 출력은 봉사기상에서 열려 진 파일을 보여 준다. 사용자는 파일로부터 열쇠 걸린 파일을 없애거나 이것들중 어느 하나를 닫을수 있다.

NET GROUP

NET GROUP−이 지령은 봉사기상에서 사용자가 그룹들을 관리하게 한다. 어떤 선택항목도 없으면 이 지령은 그룹을 표시한다.

NET GROUP 지령형식 :

C:**NET GROUP**/?

```
NET GROUP  [groupname[/COMMENT:"text"]] [DOMAIN]
           groupname{/ADD[/COMMENT:"text"] | /DELETE} [/DOMAIN]
           groupname username[…] {/ADD | /DELETE} [/DOMAIN]
```

일반적으로 사용되는 선택항목들은 다음과 같다.

/ADD 그룹을 그 그룹에 대한 령역 또는 사용자이름으로 첨가한다.

/DELETE 그룹을 그 그룹에 대한 령역 또는 사용자이름으로 삭제한다.

groupname

groupname 은 망그룹을 처리하기 위한것을 지적한다. 선택항목을 리용하지 않으면 그룹의 부분인 사용자를 표시한다. 그룹에 첨가하기 위하여 username 을 지적하기 위해서는 groupname 에 의하여 ADD 또는 DELETE 를 선택할수 있다.

/COMMENT:"text"

설명문을 groupname 에 첨가한다.

/DOMAIN 조작을 초기령역 조종기에서 실행한다. 이 선택항목은 Windows 봉사기체계에서 기정적이다.

username 사용자가 그룹에 첨가되거나 삭제된다. 임의의 사용자의 수를 지적할수 있다.

다음실레는 그룹 hp consultants 를 첨가하기 위하여 NET GROUP 지령을 사용하는것을 보여 준다.

C:/ **NET GROUP "hp consultants" /ADD**

The request will be processed at the primary domain controller for domain NSDNIS.
The comment completed successfully.
다음에는 국부체계상의 hp consultants 그룹을 위한 marty 사용자를 첨가한다.

C:/ **NET GROUP "hp consultants marty" /ADD**

The request will be processed at the primary domain controller for domain NSDNIS.

The comment completed successfully.

다음에는 hp consultants 에 대한 설명문을 첨가한다.

C:/ **NET GROUP "hp consultants" /COMMENT:" Group For HP Consultants"**

The request will be processed at the primary domain controller for domain NSDNIS.

The comment completed successfully.

이제는 hp consultants 그룹과 관련된것을 첨가한 정보를 보면서 우리가 해놓은것을 볼수 있다.

C:**NET GROUP "hp consultants"**

Group name hp consultants
Comment Group For Hp Consultants

Members

---

marty
The comment completed successfully.

이 출력은 hp consultants 그룹을 만든것을 확인하게 하며 marty 이 그룹의 성원이며 설명문이 그룹과 관련된것이라는것을 확인할수 있게 한다.

NET HELP

NET HELP−임의의 NET지령에 대한 방조를 얻기 위하여 사용한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

NET HELP command | more　　이것은 사용자가 지정한 지령에 대한 정보를 준다.

다음의 실례는 지령들의 방조목록을 보기 위하여 NET HELP 를 쓰는것을 보여 준다.

C:\ **NET HELP**

The syntax of this command is:

NET HELP command
 -or-
NET command /HELP

Commands available are:

NET ACCOUNTS	NET HELP	NET SHARE
NET COMPUTER	NET HELPMSG	NET START
NET CONFIG	NET LOCALGROUP	NETSTATISTICS
NET CONFIG SERVER	NET NAME	NET STOP
NET CONFIG WORKSTATION	NET PAUSE	NET TIME
NET CONTINUE	NET PRINT	NET USE
NET FILE	NET SEND	NET USER
NET GROUP	NET SESSION	NET VIEW

NET HELP SERVICES lists the network services you can start.
NET HELP SYNTAX explains how to read NET HELP syntax lines.
NET HELP command | MORE displays Help one screen at a time.

다음의 실례는 NET GROUP 지령에 대한 정보를 얻기 위하여 NET HELP 를 리용하는 과정을 보여 준다.

C:\ **NET HELP NET GROUP**

이 지령의 형식은 다음과 같다.

NET GROUP [groupname [/COMMENT:"text"]] [/DOMAIN]
groupname {/ADD [/COMMENT:"text"] /DELETE}
[/DOMAIN]
groupname username […] {/ADD | /DELETE} [/DOMAIN]

NET GROUP 는 봉사기에서 대역적그룹을 첨가, 표시, 변경한다. 봉사기에서 그룹이름을 표시하는 파라메터를 사용한다.

groupname	첨가, 확장, 지우기 위하여 그룹이름을 준다. 그룹에서 사용자의 목록을 보기 위하여서는 groupname 만 제시하면 된다.
/COMMENT;"text"	새것이나 존재하는 그룹에 대한 설명문을 첨가한다. 설명문은 48 개 기호까지 가질수 있다. 설명문은 인용문기호에 포함하여야 한다.
/DOMAIN	현재 령역에 대한 토대령역조종자에서 연산을 실행한다. 다른 방법으로 연산은 국부콤퓨터에서 실행한다. 이 파라메터는 Windows 봉사기령역에 대한 성원인 Windows 작업마당콤퓨터에서만 응용한다. 기정값으로는 Windows 봉사기콤퓨터들을 토대령역조종자에서 연산을 실행한다.
username[…]	그룹으로부터 첨가 또는 제거하기 위하여 한개 또는 여러개 사용자이름을 쓴다.
/ADD	그룹을 첨가하거나 그룹에 사용자이름을 첨가한다.
/DELETE	그룹을 제거하거나 그룹으로부터 사용자이름을 제거한다.
NET HELP command |	MORE displays Help 를 한 화면씩 표시한다.

NET HELPMSG

NET HELPMSG-4개수자망통보코드에 대한 정보를 제시한다. 코드에 대한 정보를 얻기 위하여서는 4개 수자를 사용한다.

NET HRLPMSG 지령의 형식 :

NET HRLPMSG message#사용자가 지정한 4 개수자통보번호에 대한 정보를 제시한다.

다음실례는 체계에 그룹을 첨가하기 위한 NET GROUP 지령의 사용과정을 보여 준다. 이 지령을 사용한후에 오유수자가 제시된다.

C:**NET GROUP "hp consultant:/ADD**
The group already exists.
More help is available by typing NET HELPMSG 2233.

4개수자코드에 대한 정보를 얻기 위하여 NET HRLPMSG를 사용할수 있다.

C:\ **NET HELPMSG 2223**

그룹이 이미 존재한다.

EXPLANATION

사용자는 이미 존재하는 그룹이름으로 그룹을 창조하려고 한다.

ACTION

새 그룹에 대하여 개별적인 그룹이름을 사용한다.
봉사기에 설정하는 그룹이름의 목록을 표시하기 위하여 사용한다.

NET GROUP

NET LOCALGROUP

NET LOCALGROUP-이 지령은 콤퓨터에서 그룹을 표시하고 관리하게 한다. 아무런 선택항목을 제시하지 않으면 그룹을 표시한다.

NET LOCALGROUP 지령형식 :

C:**NET LOCALGROUP**/?
NET LOCALGROUP [groupname[/COMMENT:"text"]] [DOMAIN]
groupname{/ADD[/COMMENT:"text"] ｜/DELETE} [/DOMAIN]
groupname username[…] {/ADD｜/DELETE} [/DOMAIN]

일반적으로 사용되는 선택항목들은 다음과 같다.

/ADD 그룹을 그 그룹에 대한 령역 또는 사용자이름으로 첨가한다.
/DELETE
그룹을 그 그룹에 대한 령역 또는 사용자이름으로 삭제한다.

groupname

조작하기 위한 groupname 을 지적한다. 선택을 하지 않으면 그룹의 부분인 사용자를 표시한다. 그룹에 첨가하기 위한 username 을 지적하기 위해서는 groupname 으로 ADD 또는 DELETE 를 선택할수 있다.

/COMMENT:"text"

설명문을 groupname 에 첨가한다.

/DOMAIN

초기령역조종기에서 조작을 실행한다. 이 선택은 Windows 봉사기체계에서 기정적이다.

name　이것은 그룹으로부터 첨가 또는 삭제하기 위한 사용자이름(s) 또는 그룹이름(s)이다. 임의의 수로 지적할수 있다.

NET NAME

NET NAME-이 지령은 콤퓨터로부터 통보이름을 첨가 또는 삭제한다. 이 지령을 NET USER 와 혼돈하지 말아야 한다. 이것은 체계상에서 사용자계산서를 첨가 또는 삭제한다.

NET NAME 지령형식 :

C:**NET NAME**/?

NET NAME [name [/ADD | /DELETE]]

일반적으로 사용되는 선택항목들은 다음과 같다.

name　　첨가 또는 삭제하기 위한 이름

/ADD　　이름을 콤퓨터에 첨가

/DELETE　　콤퓨터에서 이름을 삭제

NET PAUSE

NET PAUSE-Windows 봉사를 중지한다.

NET PAUSE 지령형식 :

C:**NET PAUSE**/?

NET PAUSE Service

사용자는 NET PAUSE 와 NET CONTINUE 지령으로 많은 Windows 봉사를 중지하거나 계속할수 있다.

아래에서는 NET LOGON 봉사를 중지하기 위하여 NET PAUSE 지령을 사용하는것을 보여 주고 NET CONTINUE 지령으로 다시 기동하는것을 보여 준다.

그림 30-3 은 체계에서 실행하는 봉사를 보여 주는 대화칸의 봉사를 체계상에서 보여 준다.

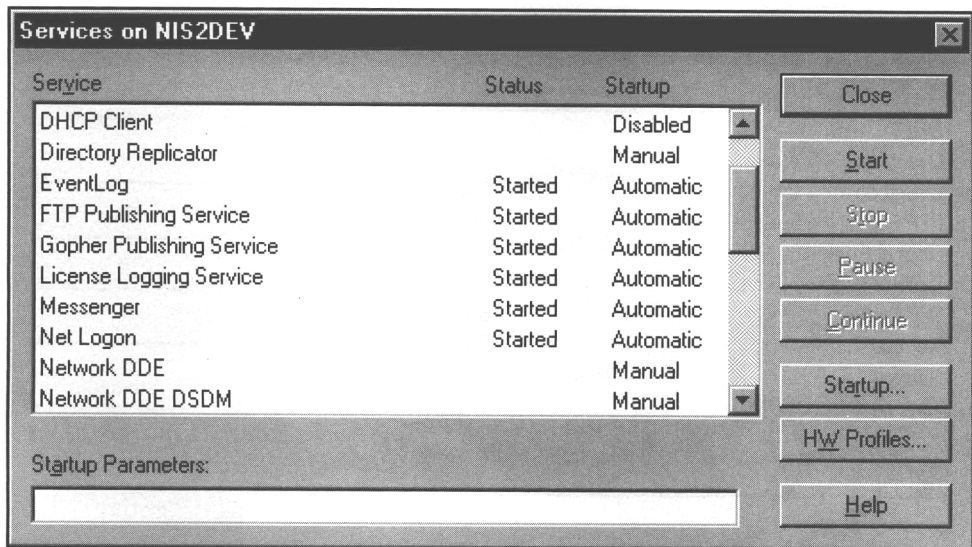



그림 30-3. Net Logon 이 시작된 대화칸봉사

NET LOGON 봉사를 중지하기 위하여 NET PAUSE 지령을 사용한다.

C:**NET PAUSE NTLOGON**
The Net Logon service was paused successfully.

그림 30-4 는 대화칸봉사를 반복하여 보여 준다. 이번에는 NET LOGON 봉사가 중지되였다는것을 알수 있다.

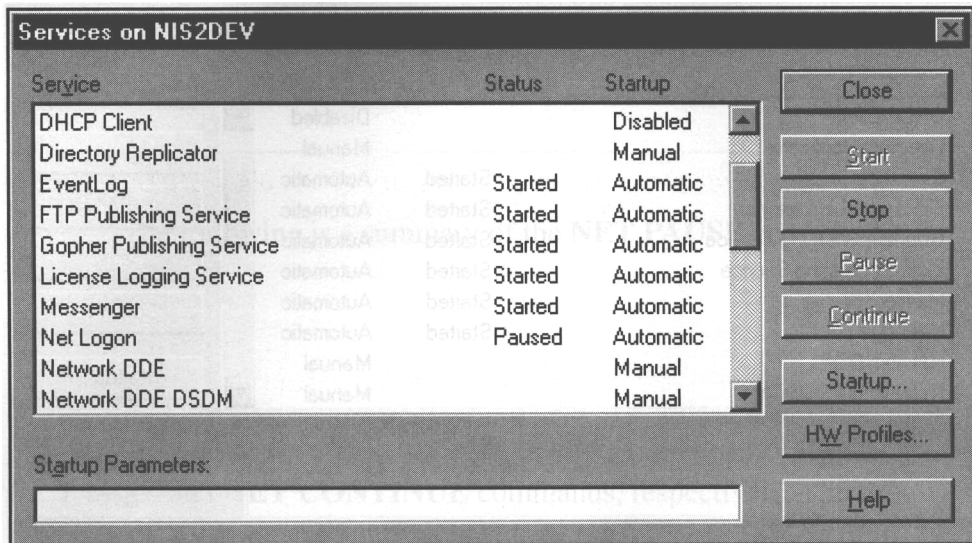



그림 30-4. Net Logon 이 중지된 대화칸봉사

다음으로 NET CONTINUE 지령으로 NET LOGON 봉사를 계속할수 있다.

C:**NET CONTINUE NTLOGON**
The Net Logon service was continued successfully.

NET PRINT

NET PRINT-이 지령은 인쇄과제나 공유대기렬을 표시한다.

NET PRINT 지령형식 :

C:**NET PRINT**/?

NET PRINT \\computername\sharename
[\\computername] job#[/HOLD | /RELEASE | /DELETE]

일반적으로 사용하는 선택항목들은 다음과 같다.

\\computername	computername 은 인쇄대기렬을 공유한다.
sharename	sharename 은 인쇄대기렬이다.
job#	인쇄과제에 주는 대기렬번호이다.
/HOLD	과제는 HOLD 에 대한 상태를 제시한다. 이것은 해체 또는 인쇄되지 않는다는것을 의미한다.
/RELEASE	인쇄될수 있도록 인쇄될 과제를 해체한다.
/DELETE	인쇄대기렬에서 인쇄될 과제를 지운다.

NET SEND

NET SEND−망상에서 다른 사용자, 콤퓨터, 통보이름들의 통보를 전송한다.

NET SEND 지령형식 :

C:**NET SEND**/?

NET SEND {name | * | /DOMAIN[:name] | /USERS} message

일반적으로 사용되는 선택항목들은 다음과 같다.

name 전송될 사용자이름, 콤퓨터이름, 통보이름이다.

* 사용자그룹을 개별적인 사용자들에게보다도 모든 사용자들에게 통보를 보내기 위하여 *이 사용된다.

/DOMAIN［:domainname］
/DOMAIN 에서 모든 사용자에게 통보를보내기 위해 /DOMAIN 을 사용한다. 사용자가 통보전송을 요구하는 domainname 을 지정할수 있다.

message 이것은 사용자가 보낼줄것을 요구하는 본문통보이다.

다음 실례는 개별적인 사용자에게 통보를 보내기 위한 NET SEND 지령을 사용한데 대하여 보여 준다.

NET SEND marty our NetServer LXr has arrived for installation.

그림 30-5 는 marty 가 작업하고 있는 콤퓨터의 화면에 나타난 경보창문을 보여 준다.

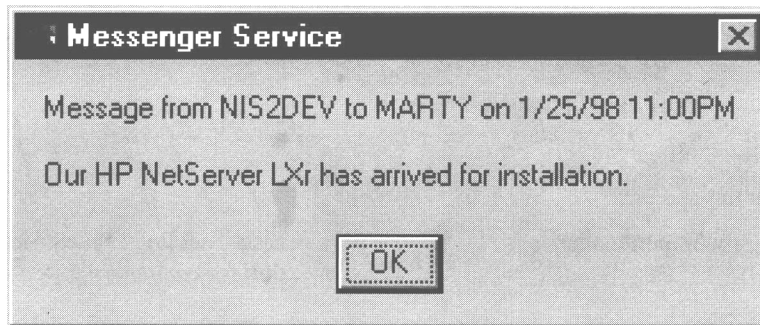



그림 30-5. NET SEND지령으로 산생된 경보창문

NET SESSION

NET SESSION-콤퓨터들사이에 대화시간을 설정하거나 단절하는것을 설정한다.

NET SESSION지령형식 :

C:**NET SESSION**/?
NET SESSION [\\computername] [/DELETE]

일반적으로 사용되는 선택항목들은 다음과 같다.

\\computername	computername에 대화시간통보를 표시한다.
/DELETE	국부콤퓨터와 computername 사이의 대화시간을 끝내거나 열려 진 파일을 닫으면서 computername 이 없이 모든 대화시간을 끝낸다.

NET SHARE

NET SHARE-이 지령은 다른 망사용자들에게 봉사기자원을 공유하거나 이미 존재하는 공유자원들에 대한 정보를 렬거하는데 리용된다. 사용자가 어떤 선택항목도 지적하지 않으면 지령은 이미 존재하는 공유자원들에 대한 정보를 렬거한다.

NET SHARE 지령형식 :

C: \ **NET SHARE /?**

```
NET SHARE sharename
        sharename=drive:path    [USERS:number] /UNLIMITED]
                                [REMARK:"text"]
        sharename  [/USERS:number | /UNLIMITED]
                   [/REMARK:"text"]l
        {sharename | devicename | drive:path} /DELETE
```

일반적으로 사용하는 선택항목들은 다음과 같다.

sharename	공유된 자원의 망의 이름이다. sharename 만 리용하면 공유자원에 대한 정보를 표시한다.
devicename	sharename 으로 공유된 인쇄기들을 지정한다. devicename 으로 LP1-LPT9 를 리용한다.
drive:path	이 선택항목은 등록부에 대한 구동기와 경로 를 공유하는데 리용된다.
/USERS:number	공유된 자원을 동시에 처리할수 있는 사용자의 최대수를 지정한다.
/UNLIMITED	사용자들의 수를 제한하지 않으면 동시에 공유된 원천을 처리한다.
/REMARK:"text"	지적된 공유자원의 재표시와 관련된것이다.
/DELETE	공유를 해제한다.

체계에 새로운 공유를 설치하기 위하여 NET SHARE 를 사용한다.

아래에서 시작하기전에 선택항목을 설정하지 않고 NET SHARE 지령을 리용하여 이미 존재하는 공유자원을 보면 다음과 같다.

```
D:\ NET SHARE

Share name      Resource                          Remark
-----------------------------------------------------------------------------
ADMIN$          D:\WINNT                          Remote Admin
IPC$                                              Remote IPC
C$              C:\                               Default share
print$          D:\WINNT\system32\spool\drivers   Printer Drivers
D$              D:\                               Default share
E$              E:\                               Default share
net_share       C:\net_share
```

NETLOGON	D:\WINNT\system32\Repl\Import\S		Logon server share
HPLaserJ5	LPT1:	Spooled	HP LaserJet 5MP

The command completed successfully.

이제는 measure 라는 하나의 공유이름과 최대로 5 명의 사용자들로 C:\measureware 등록부의 공유를 다음과 같이 설치할수 있다.

C:\NET SHARE measure=c:\measure ware/users:5
measure was shared successfully

NET SHARE 지령은 공유이름이 "measureware"로 되였다는것을 다음과 같이 보여준다.

C:\ NET SHARE

Share name	Resource	Remark
D$	D:\	Default share
ADMIN$	D:\WINNT	Remote Admin
IPC$		Remote IPC
C$	C:\	Default share
print$	D:\WINNT\system32\spool\drivers	Printer Drivers
E$	E:\	Default share
measure	C:\measure ware	
net_share	C:\net_share	
NETLOGON	D:\WINNT\system32\Repl\Import\S	Logon server share
HPLaserJ5	LPTI: Spooled	HP LaserJet 5MP

The command completed successfully.

NET START

NET START-이 지령은 NET STOP 지령을 사용하여 정지되였던 봉사를 시작하거나 시작하지 않은 봉사를 시작한다. 현재 실행되고 있는 봉사들의 목록을 보기 위해서는 선택항목이 없이 NET START를 쓴다.

NET START 지령형식 :

C:\ **NET START**/?
NET START [service]

다음실례는 선택항목을 가지지 않는 NET START 지령을 리용하여 현재 실행하는 봉사에 대한 표시를 보여 준다.

```
C:/ NET START

These Windows services are started:

Alerter
    Computer Browser
    EventLog
    FTP Publishing Service
    Gateway Service for NetWare
    Gopher Publishing Service
    License Logging Service
    MeasureWare Agent
    MeasureWare Transaction Manager
    Messenger
    Net Logon
    NT LM Security Support Provider
    OracleServiceTMI
    OracleStartTMI
    OracleTNSListener
    Plug and Play
    Remote Procedure Call（RPC）Locator
    Remote Procedure Call（RPC）Service
    Schedule
    Server
    Spooler
    STP/A - TCP/IP Page Server（TMI）
    TCP/IP NetBIOS Helper
    UPS
    Workstation
    World Wide web Publishing Service

The command completed successfully.
```

실례에는 창문에 나타나지 않은 많은 봉사목록이 있다. 기타 Measure Ware(Hp 수행도구들)과 오라클(oracle)봉사들은 각각 NET START 와 NET STOP 지령들에 의하여 시작될수 있거나 정지될수 있다.

NET STATISTICS

NET STATISTICS-이 지령은 봉사의 **통계일지**(Statistics Log)를 표시하기 위하여 사용된다. 선택항목이 없이 NET STATISTICS 를 쓰면 모든 봉사들에 대한 통계를 표시한다.

NET STATISTICS 지령형식 :

C:\ **NET STATISTICS**/?

NET STATISTICS [WORKSTATION | SERVER]

일반적으로 사용되는 선택항목들은 다음과 같다.

SERVER	봉사기에 대한 통계봉사를 표시한다.
WORKSTATION	워크스테이션의 통계봉사를 표시한다.

다음 실례는 선택항목이 없는 NET STATISTICS 지령을 수행하는 과정을 보여 준다.

```
C:\ NET STATISTICS
Server Statistics for \\NISDEV

Statistics since 1/21/98 8:49 AM

Sessions accepted              3
Sessions timed-out             32
Sessions errored-out           32

Kilobytes sent                 165941
Kilobytes received             33118

Mean response time (msec)      0

System errors                  0
Permission violations          2
Password violations            0

Files accessed                 12787
Communication devices accessed 0
Print jobs spooled             0

Times buffers exhausted

Big buffers                    0
Request buffers                0

The command completed successfully.
```

NET STOP

NET STOP-이 지령은 NET START지령을 사용하여 시작된 봉사를 정지한다.

NET STOP 지령형식 :

C:**NET STOP**/?
NET STOP service

NET STOP지령의 한개 선택항목의 의미는 다음과 같다.

service　　　　정지할수 있는 봉사의 이름이다.

NET TIME

NET TIME-이 지령은 콤퓨터시간을 표시하거나 혹은 어느한 콤퓨터의 시간을 다른 콤퓨터의 시간과 정합을 위하여 리용된다.

NET TIME 지령형식 :

C:**NET TIME**/?
NET TIME [\\computername | /DOMAIN[:domainname]] [/SET]

일반적으로 사용되는 선택항목들은 다음과 같다.

\\ computername　　　시간을 맞추거나 시간을 표시하려는 콤퓨터의 이름이다.

/DOMAIN[:domainname]
　　　시간을 맞추려는 령역의 이름이다.

/SET　　　지적된 콤퓨터나 령역의 시간으로 콤퓨터시간을 설정한다.

NET USE

NET USE-콤퓨터의 접속을 표시하거나 공유된 자원들을 없애거나 허용하게 한다. 선택항목이 없으면 이 지령은 콤퓨터의 접속을 표시한다.

NET USE 지령형식 :

C:\ **NET USE** /?
NET USE [devicename|*] [//computername\sharename[/volume] [password | *]]

[/USER:[domainname\]username]
[[/DELETE] | [/PERSISTENT:{YES | NO}]]

NET USE [devicename |*][[password |*]][/HOME]

NET USE [PERSISTENT:{YES | NO}]

일반적으로 사용되는 선택항목들은 다음과 같다.

devicename	접속하거나 접속을 끊기 위한 이름을 지적한다. 이름으로는 구동기나 인쇄기와 같은 이름을 줄수 있다.
//computername	이것은 공유자원들을 조종하는 콤퓨터의 이름이다.
/sharename	공유된 장치들의 망이름이다.
password	공유된 자원들에 접근하기 위한 암호이다.
*	공유된 자원을 사용하기 위해 사용자들을 위한 통과단어의 입력재촉문이 나타나게 한다.
/USER	접속하기 위한 서로 다른 사용자이름을 지적한다.
domainname	현재 령역으로부터 다른 령역이 사용된다.
username	사용자가입이름을 지적한다.
/HOME	사용자와 그들의 홈등록부를 접속한다.
/DELETE	일정한 접속목록으로부터 망접속을 취소하고 그것을 없앤다.
/PERSISTENT{yes/no}	yes 는 접속이 다음 가입에서 재수립된다는것을 의미한다. no 는 앞으로 가입하기 위한 접속을 보관하지 않는다는것을 지적한다.

다음 실례는 선택항목이 없는 NET USE 지령의 수행과정을 보여 준다.

```
C:\ NET USE

New connections will be remembered.

Status    Local   Remote              Network
------------------------------------------------------------------------------------------------
OK        P:      \\DEVSYS\Disk_C     Microsoft Windows Network
The command completed successfully.
```

원격체계 DEVSYS 우의 Disk_C 등록부를 국부체계 P:안에서 보여 준다.

NET USER

NET USER – 이 지령은 콤퓨터계산서를 창조하거나 수정한다. 이 지령에서 선택항목을 쓰지 않으면 계산서목록만 보여 준다.

NET USER 지령형식 :

C:**NET USER**/?

```
NET USER    [username [password | *] [options]] [/DOMAIN]
            username {password | *} /ADD [options] [/DOMAIN]
            username [/DELETE] [/DOMAIN]
```

일반적으로 사용되는 선택항목들은 다음과 같다.

/ADD	사용자계산서를 첨가한다.
/DELETE	사용자계산서를 삭제한다.
username	관리하기 위한 계산서의 이름이다. 사용자는 첨가, 삭제, 수정, 계산서보기를 할수 있다.
Password	사용자계산서의 암호를 할당하거나 교체한다.
*	암호대기를 표시한다.
/DOMAIN	작용이 초기령역조종자로서 실행된다.
보충적인 선택항목들	계산서의 날자표시, 사용자의 가입환경의 경로 그리고 다른것들에 대한 보충적인 선택항목도 있다.

다음 실례는 선택항목이 없는 NET USER 지령의 수행과정을 보여 준다.

```
C:\ NET USER

User accounts for \\NIS2DEV
-------------------------------------------------------------------------------
Administrator         arleo_j            burgos_c
Guest                 IUSR_NIS2DEV       IUSR_NISDEV
johnt                 leung_k            marty
mckenna_b
The command completed successfully.
```

앞에서 본 지령의 두번째 형식을 리용하여 체계에 새로운 사용자를 첨가할수 있다.

C:\ NET USER amyp * /ADD

Type a password for the user:

Retype the password to confirm:

The request will be processed at the primary domain controller for domain DEV.

The command completed successfully.

NET USER 지령가운데서 *는 새로운 사용자 amyp 에 대한 암호를 재촉하는 부호로 된다는것을 의미한다.

NET VIEW

NET VIEW - 이 지령은 콤퓨터상에서 공유되여 있는 자원을 표시한다. 선택항목이 없이 쓰면 령역의 콤퓨터목록을 표시한다.

NET VIEW지령형식 :

C:**NET VIEW**/?

NET VIEW [\\computername | /DOMAIN[:domainname]]
NET VIEW /NETWORK:NW [\\computername]

일반적으로 사용되는 선택항목들은 다음과 같다.

//computer name 공유된 자원을 보려고 하는 콤퓨터이름을 지적한다.

/DOMAIN:domainname
공유된 자원을 보려고 하는 령역의 이름을 지적한다.

/NETWORK:NW 망에서 망봉사기를 표시한다.

다음실례는 현재 령역에서 콤퓨터목록을 산생시키기 위하여 선택항목이 없는 NET VIEW지령을 리용하는것을 보여 준다.

C:\ NET VIEW

Server Name Remark

\\NIS2DEV
\\NISDEV
The command completed successfully.

다음의 실례는 DOMAIN에서 NET VIEW명령문을 리용하는것을 보여 준다.

```
C:\ NET VIEW /DOMAIN:nisdomain

Server Name        Remark
-------------------------------------------------------------------------------------------------
\\HPSYSTEM1
\\KITTY
\\NIS
The command completed successfully.
```

다음실례는 현재 령역에서 개별적인 computername을 기록하는 NET VIEW지령을 리용하는것을 보여 준다.

```
C:\ NET VIEW \\nisdev
Shared resources at \\hpsysteml

Share name       Type      Used as  Comment
-------------------------------------------------------------------------------------------------
HPLaserJ5        Print              HP LaserJet 5MP
Measure          Disk
net_share        Disk
NETLOGON         Disk               Logon server share
The command completed successfully.
```

POSIX 봉사프로그람

Microsoft Windows NT 봉사기 Resource Kit(이 장에서 Resource Kit로 지정한)는 UNIX 체계관리자가 Windows를 사용할 때 편리하게 리용하는 여러개의 POSIX 봉사프로그람을 가지고 있다. Resource Kit는 일반적으로 복잡한 체계관리자원이다. 이 책에서의 POSIX 봉사프로그람들만 고찰해도 Resource Kit의 내용은 적지 않게 알수 있다. Resource Kit는 Microsoft Press, Redmond, WA에서 리용된다. POSIX 봉사프로그람들은 cat, chmod, find, ls, mv 그리고 다른것들과 같은 워크스테이션지령들을 포함하고 있다. Resource Kit에서 효과적인 지령들은 구조에 따라 약간씩 차이난다. 이 장에서는 "I386" 봉사프로그람만 지적하고 그것과 다르게 설계된 봉사프로그람들은 고찰하지 않는다.

Resource Kit는 그우에 POSIX.WRI 파일을 가지고 있으므로 POSIX 봉사를 구체적으로 서술한다. 이 장에서는 간단한 봉사프로그람개괄과 봉사프로그람을 사용한 실례를 제시한다. 사용자는 MicrosoftWeb 싸이트에서 Resource Kit에 대한 더 많은 정보를 볼수 있다.

Windows 봉사기와 워크스테이션방안 역시 둘 다 Resource Kit 에 있다. 이 장에 있는 POSIX 의 지령들은 봉사기 Resource Kit 로 사용된다. POSIX 봉사프로그람을 위한 원천코드와 실행자는 둘 다 Resource Kit 에 있다. 아래에서는 Resource Kit CD-ROM 에서 I386 을 위한 POSIX 실행목록에 대하여 보여 주고 있다.

```
F:\I386\GNUJ\POSIX> ls -1
-rwxrwxrwx  1  Everyone  Everyone  101748  Sep  6  12:39  CAT.EXE
-rwxrwxrwx  1  Everyone  Everyone  116188  Sep  6  12:39  CHMOD.EXE
-rwxrwxrwx  1  Everyone  Everyone  110920  Sep  6  12:39  CHOWN.EXE
-rwxrwxrwx  1  Everyone  Everyone  111208  Sep  6  12:39  CP.EXE
-rwxrwxrwx  1  Everyone  Everyone  173580  Sep  6  12:39  FIND.EXE
-rwxrwxrwx  1  Everyone  Everyone  144256  Sep  6  12:39  GREP.EXE
-rwxrwxrwx  1  Everyone  Everyone   90960  Sep  6  12:39  LN.EXE
-rwxrwxrwx  1  Everyone  Everyone  128532  Sep  6  12:39  LS.EXE
-rwxrwxrwx  1  Everyone  Everyone   88984  Sep  6  12:39  MKDIR.EXE
-rwxrwxrwx  1  Everyone  Everyone   99096  Sep  6  12:39  MV.EXE
-rwxrwxrwx  1  Everyone  Everyone  114564  Sep  6  12:39  RM.EXE
-rwxrwxrwx  1  Everyone  Everyone   85004  Sep  6  12:39  RMDIR.EXE
-rwxrwxrwx  1  Everyone  Everyone  362528  Sep  6  12:39  SH.EXE
-rwxrwxrwx  1  Everyone  Everyone   91244  Sep  6  12:39  TOUCH.EXE
-rwxrwxrwx  1  Everyone  Everyone  287628  Sep  6  12:39  VI.EXE
-rwxrwxrwx  1  Everyone  Everyone   95392  Sep  6  12:39  WC.EXE
```

이 봉사프로그람이 배치된 등록부는 F 구동기인데 그 구동기는 CD-ROM 에서 I386 ￥ GNU ￥ POSIX 에 있는 I386 봉사프로그람방안이다.

그 다음의것들은 POSIX 봉사프로그람의 지령개요들이다. 이것은 봉사프로그람에 대하여 일반적으로 사용된 선택항목들과 봉사프로그람에 대한 간단한 설명이다. Resource Kit 의 POSIX.WRI 파일은 매 봉사프로그람에 대한 실례를 준다.

cat

cat – 파일을 표시, 결합, 첨가, 복사 또는 창조한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

- -n 출구행에 따라 행번호를 표시한다.
- -u 완충마당이 없이 출구하며 이 선택항목은 매 문자를 조종한다는것을 의미한다.
- -v 대부분 출구되지 않은 기호들을 시각적으로 보여 준다.

다음 실례는 cat에서 선택항목 -n의 사용에 대하여 보여 준다.

```
D:\WINNT\system> cat -n setup.inf

 1  [setup]
 2      help = setup.hlp
 3
 4  ;   Place any programs here that should be run at the end of setup.
 5  ;   These apps will be run in order of their appearance here.
 6  [run]
 7
 8  [dialog]
 9      caption    = "Windows Setup"
10      exit       = "Exit Windows Setup"
11      title      = "Installing Windows 3.1"
12      options    = "In addition to installing Windows 3.1, you can:"
13      printwait  = "Please wait while Setup configures your printer(s) …"
              ⌇
20  [data]
21  ;   Disk space required
22  ;   <type of setup>= <Full install space>, <Min install space>
23
24      upd2x386full = 10000000, 6144000 ; 10.0 Mb,  6.144 Mb
25      upd2x286full = 9000000, 6144000  ;  9.0 Mb,  6.144 Mb
26      upd3x386full = 5500000, 5000000  ;  5.5 Mb,    5.0 Mb
27      upd3x286full = 5500000, 5000000  ;  5.5 Mb,    5.0 Mb
28
29      new386full = 10000000, 6144000   ; 10.0 Mb,  6.144 Mb
30      new286full = 9000000, 6144000    ;  9.0 Mb,  6.144 Mb
31
32      netadmin = 16000000              ; 16.0 Mb
33      netadminupd = 16000000           ; 16.0 Mb
34      upd2x386net = 300000             ;   .3 Mb
35      upd3x386net = 300000             ;   .3 Mb
36      upd2x286net = 300000             ;   .3 Mb
37      upd3x286net = 300000             ;   .3 Mb
38      new386net = 300000, 300000       ;   .3 Mb,     .3 Mb
39      new286net = 300000, 300000       ;   .3 Mb,     .3 Mb
40
```

41
42
43 ; Defaults used in setting up and names of a few files
44 startup = WIN. COM

chmod

chmod – 기호방식 또는 절대적방식(때때로 수자적호출이라고 부른다.)을 사용하여 렬거된 파일들의 허락을 변경시킨다.

기호방식은 아래에 서술된다.

영향을 주는 기호 :

u 사용자가 영향을 받는다.
g 그룹이 영향을 받는다.
o 다른것들이 영향을 받는다.
a 모든 사용자가 영향을 받는다.

실행을 위한 연산 :

+ 허락을 첨가한다.
- 허락을 삭제한다.
= 허락을 재배치한다.

허락지정 :

r 허락을 읽는다.
w 허락을 쓴다.
x 허락을 실행한다.
u 사용자허락을 복사한다.
g 그룹허락을 복사한다.
o 다른 허락을 복사한다.

다음실례는 두가지 방식을 리용한다. 절대적방식 또는 수자호출을 사용하여 cat1. exe 파일에서 허락을 667 로부터 777 로 변환한다. 실행허락은 모든 사용자들을 위하여 기호방식의 리용이 제거될 때 진행된다.

D:\> **ls -1 catl. exe**
-rw-rw-rw- 1 Administ Administ 71323 Feb 20 11:34 cat1. exe

D:\> **chmod 777 catl. exe**

D:\> **ls -1 catl. exe**

```
-rwxrwxrwx 1    Administ   Administ   71323    Feb  20   11:34   catl.exe
D:\> chmod a-x catl.exe
D:\> ls -1 catl.exe
-rw-rw-rw-   1    Administ   Administ   71323    Feb  20   11:34   catl.exe
```

cp

cp － 파일들과 등록부들을 복사한다.

일반적으로 리용되는 선택항목들은 다음과 같다.

-i	이미 존재하는 파일에 덧쓰기를 하겠는가 하는 조건에 따라 복사한다.
-f	이미 존재하는 파일이름이 이미 지적되여 있으면 복사되여 있는 파일에 덧쓰기를 한다.
-p	복사할 때 허락을 보호한다.
-R	부분나무를 포함하면서 반복복사한다.

다음의 실례는 cat1.exe 를 cat2.exe 로 복사하기 위하여 cp 지령을 사용하는것을 보여준다. 이때 cat 로 시작하는 모든 파일목록이 나타난다.

```
D:\> cp catl.exe cat2.exe
D:\> ls -1 cat*

-rw-rw-rw-   1    Administ   Administ   71323    Feb  20   11:34   catl.exe
-rw-rw-rw-   1    Administ   Administ   71323    Feb  20   11:47   cat2.exe
```

find

find － 등록부구조를 반복적으로 내리훑으면서 표시된 파일을 찾는다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-f	가로찾기하는 파일조직을 한다.
+-s	기호적련결을 진행할 때 련결에 참조된 파일과 련결되지 않는것들은 자체로 리용된다.

-x 내려훑기 시작한 파일과 다른 장치번호를 가지는 등록부를 내리훑는다.

-print 표준출구를 위한 경로이름을 출구한다.

-size n 파일크기가 n이면 참이다.

grep

grep - 본문이나 표시결과를 탐색한다.

다음실례는 setup.inf 파일안에서 단어 "shell"이 있는 모든 위치를 찾기 위하여 grep을 리용하는것을 보여 준다.

D:\> **grep shell setup.inf**

```
[shell]
00000000 = "shell versions below 3.01", , unsupported_net
00030100 = "shell versions below 3.21", , novell301
00032100 = "shell versions 3.21 and above", , novell321
00032600 = "shell versions 3.26 and above", , novell326
    #win.shell, 0:
    #win.shell, 0:
[win.shell]
    shell.dll
system.ini, Boot, "oldshell"      , "shell"
```

ls

ls -등록부내용을 표시한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-a 모든 등록부를 표시한다.

-c 표시된 파일들을 정돈하기 위하여 마지막으로 수정된 파일의 시간을 리용한다.

-d 등록부의 내용은 없이 이름만을 표시한다.

-g 출구에 그룹을 포함한다.

-i 보고서의 첫렬에 매듭번호를 출구한다.

-q "?"에 의해서 인쇄되지 않은 문자들이 표시된다.

-r 파일이 인쇄되는 순서를 뒤집는다.

-s 바이트대신에 블로크로 크기를 나타낸다.

-t 가장 최근에 기억한 시간순서로 표시한다.

-u　　표시된 파일의 순서를 확정하기 위하여 마지막으로 수정된것대신에 마지막으로 처리된 시간을 리용한다.

-A　　-a 와 같은데 표시되지 않은 현재 등록부와 그리고 이 등록부와 관계된 어미등록부를 표시하지 않는다.

-C　　다중행출구를 한다.

-F　　"/"은 등록부, "*"은 수행, "@"은 기호적련결을 의미한다.

-L　　파일과 등록부의 련결점을 표시한다.

-R　　보조등록부를 반복표시한다.

이와 관련한 몇가지 실례들을 제시하면 다음과 같다.

D:\　**ls -a**

Blue Monday 16.bmp
Blue Monday.bmp
Coffee Bean 16.bmp
Coffee Bean.bmp
Config
Cursors
FORMS
FeatherTexture.bmp
Fiddle Head.bmp
Fonts
Furry Dog 16.bmp
Furry Dog.bmp
Geometrix.bmp
Gone Fishing.bmp
Greenstone.bmp
Hazy Autumn 16.bmp
Help
Hiking Boot.bmp
Leaf Fossils 16.bmp
Leather 16.bmp
Maple Trails.bmp
Media
NETLOGON.CHG
NOTEPAD.EXE
Petroglyph 16.bmp
Prairie Wind.bmp
Profiles

REGEDIT. EXE
Rhododendron. bmp
River Sumida. bmp
Santa Fe Stucco. bmp
Seaside 16. bmp
Seaside. bmp
ShellNew
Snakeskin. bmp
Soap Bubbles. bmp
Solstice. bmp
Swimming Pool. bmp
TASKMAN. EXE
TEMP
Upstream 16. bmp
WIN. INI
WINFILE. INI
WINHELP. EXE
Zapotec 16. bmp
Zapotec. bmp
_DEFAULT. PIF
blackl6. scr
clock. avi
control. ini
explorer. exe
inetsrv. mif
inf
lanma256. bmp
lanmannt. bmp
network. wri
poledit. exe
printer. wri
repair
setup. old
setuplog. txt
system
system. ini
system32
vmmreg32. dll
welcome. exe
winhlp32. exe

D:\> **ls ?1**

```
-rwxrwxrwx   1  Administ  NETWORK    8310  Aug   9   1996  Blue Monday  16.bmp
-rwxrwxrwx   1  Administ  NETWORK   37940  Aug   9   1996  Blue Monday.bmp
-rwxrwxrwx   1  Administ  NETWORK    8312  Aug   9   1996  Coffee Bean 16.bmp
-rwxrwxrwx   1  Administ  NETWORK   17062  Aug   9   1996  Coffee Bean.bmp
drwx---rwx   1  Administ  Administ      0  Feb  10  10:39  Config
drwx---rwx   1  Administ  Administ      0  Feb  10  16:22  Cursors
drwxrwxrwx   1  Administ  NETWORK       0  Feb  10  16:23  FORMS
-rwxrwxrwx   1  Administ  NETWORK   16730  Aug   9   1996  FeatherTexture.bmp
-rwxrwxrwx   1  Administ  NETWORK   65922  Aug   9   1996  Fiddle Head.bmp
drwx --- rwx 1  Administ  Administ   8192  Feb  10  10:39  Fonts
-rwxrwxrwx   1  Administ  NETWORK   18552  Aug   9   1996  Furry Dog 16.bmp
-rwxrwxrwx   1  Administ  NETWORK   37940  Aug   9   1996  Furry Dog.bmp
-rwxrwxrwx   1  Administ  NETWORK    4328  Aug   9   1996  Geometrix.bmp,
-rwxrwxrwx   1  Administ  NETWORK   17336  Aug   9   1996  Gone Fishing.bmp
-rwxrwxrwx   1  Administ  NETWORK   26582  Aug   9   1996  Greenstone.bmp
-rwxrwxrwx   1  Administ  NETWORK   32888  Aug   9   1996  Hazy Autumn 16.bmp
drwx --- rwx 1  Administ  Administ      0  Feb  19  15:10  Help
-rwxrwxrwx   1  Administ  NETWORK   37854  Aug   9   1996  Hiking Boot.bmp
-rwxrwxrwx   1  Administ  NETWORK   12920  Aug   9   1996  Leaf Fossils 16.bmp
-rwxrwxrwx   1  Administ  NETWORK    6392  Aug   9   1996  Leather 16.bmp
-rwxrwxrwx   1  Administ  NETWORK   26566  Aug   9   1996  Maple Trails.bmp
drwx --- rwx 1  Administ  Administ      0  Feb  10  16:23  Media
-rwxrwxrwx   1  Administ  NETWORK   65536  Feb  11  10:35  NETLOGON.CHG
-rwxrwxrwx   1  Administ  NETWORK   45328  Aug   8   1996  NOTEPAD.EXE
-rwxrwxrwx   1  Administ  NETWORK   16504  Aug   9   1996  Petroglyph 16.bmp
-rwxrwxrwx   1  Administ  NETWORK   65954  Aug   9   1996  Prairie Wind.bmp,
drwxrwxrwx   1  Administ  NETWORK    4096  Feb  10  16:32  Profiles
-rwxrwxr-x   1  Administ  NETWORK   71952  Aug   8   1996  REGEDIT.EXE
-rwxrwxrwx   1  Administ  NETWORK   17362  Aug   9   1996  Rhododendron.bmp
-rwxrwxrwx   1  Administ  NETWORK   26208  Aug   9   1996  River Sumida.bmp
-rwxrwxrwx   1  Administ  NETWORK   65832  Aug   9   1996  Santa Fe Stucco.bmp
-rwxrwxrwx   1  Administ  NETWORK    8312  Aug   9   1996  Seaside 16.bmp
-rwxrwxr-x   1  Administ  NETWORK   17334  Aug   9   1996  Seaside.bmp
drwxrwxrwx   1  Administ  NETWORK       0  Feb  10  16:22  ShellNew
-rwxrwxrwx   1  Administ  NETWORK   10292  Aug   9   1996  Snakeskin.bmp
-rwxrwxrwx   1  Administ  NETWORK   65978  Aug   9   1996  Soap Bubbles.bmp
-rwxrwxr-x   1  Administ  NETWORK   17334  Aug   9   1996  Solstice.bmp
```

```
-rwxrwxrwx   1  Administ  NETWORK    26202  Aug   9   1996  Swimming Pool.bmp
-rwxrwxrwx   1  Administ  NETWORK    32016  Aug   8   1996  TASKMAN.EXE
drwxrwxrwx   1  Administ  NETWORK        0  Feb  20  09:59  TEMP
-rwxrwxrwx   1  Administ  NETWORK    32888  Aug   9   1996  Upstream 16.bmp
-rwxrwxrwx   1  Administ  NETWORK      239  Feb  10  16:23  WIN.INI
-rwxrwxr-x   1  Administ  NETWORK        3  Aug   8   1996  WINFILE.INI
-rwxrwxr-x   1  Administ  NETWORK   256192  Aug   8   1996  WINHELP.EXE
-rwxrwxrwx   1  Administ  NETWORK     8312  Aug   9   1996  Zapotec 16.bmp
-rwxrwxr-x   1  Administ  NETWORK     9522  Aug   9   1996  Zapotec.bmp
-rwxrwxr-x   1  Administ  NETWORK      707  Aug   8   1996  DEFAULT.PIF
-rwx --- r-x 1  Administ  Administ    5328  Aug   8   1996  Elack16.scr
-rwx --- r-x 1  Administ  Administ   82944  Aug   8   1996  clock.avi
-rwxrwxrwx   1  Administ  NETWORK        0  Feb  10  11:18  control.ini
-rwx --- r-x 1  Administ  Administ  234256  Aug   8   1996  explorer.exe
-rwxrwxrwx   1  Administ  NETWORK     1628  Feb  10  11:20  inetsrv.mif
drwx --- rwx 1  Administ  Administ   47104  Feb  10  10:56  inf
-rwx --- r-x 1  Administ  Administ  157044  Aug   8   1996  lanma256.bmp
-rwx --- r-x 1  Administ  Administ  157044  Aug   8   1996  lanmannt.bmp
-rwx --- r-x 1  Administ  Administ   67328  Aug   8   1996  network.wri
-rwx --- r-x 1  Administ  Administ  123152  Aug   8   1996  poledit.exe
-rwx --- r-x 1  Administ  Administ   34816  Aug   8   1996  printer.wri
drwx --- rwx 1  Administ  Administ       0  Feb  10  16:24  repair
-rwxrwxrwx   1  Administ  NETWORK     2499  Feb  10  16:23  setup.old
-rwxrwxrwx   1  Administ  NETWORK      138  Feb  10  16:22  setuplog.txt
drwx---rwx   1  Administ  Administ    4096  Feb  20  10:07  system
-rwx --- r-x 1  Administ  Administ     219  Aug   8   1996  system.ini
drwx --- rwx 1  Administ  Administ  167936  Feb  20  09:50  system32
-rwx --- r-x 1  Administ  Administ   24336  Aug   8   1996  vmmreg32.dll
-rwx --- r-x 1  Administ  Administ   22288  Aug   8   1996  welcome.exe
-rwx --- r-x 1  Administ  Administ  310032  Aug   8   1996  winhlp32.exe
```

D:\> **ls ?C**

```
Blue Monday 16.bmp  Greenstone.bmp       Rhododendron.bmp     WINFILE.INI     poledit.exe
Blue Monday.bmp     HazyAutumn 16.bmp    River Sumida.bmp     WINHELP.EXE     printer.wri
Coffee Bean 16.bmp  Help                 Santa Fe Stucco.bmp  Zapotec 16.bmp  repair
Coffee Bean.bmp     Hiking Boot.bmp      Seaside 16.bmp       Zapotec.bmp     setup.old
Config              Leaf Fossils 16.bmp  Seaside.bmp _DEFAULT.PIF             setuplog.txt
Cursors             Leather 16.bmp       ShellNew             black16.scr     system
```

FORMS	Maple Trails.bmp	Snakeskin.bmp	clock.avi	system.ini
FeatherTexture.bmp	Media	Soap Bubbles.bmp	control.ini	system32
Fiddle Head.bmp	NETLOGON.CHG	Solstice.bmp	explorer.exe	vmnmreg32.dll
Fonts	NOTEPAD.EXE	Swimming Pool.bmp	inetsrv.mif	welcome.exe
Furry Dog 16.bmp	Petroglyph 16.bmp	TASKMAN.EXE	inf	winhlp32.exe
Furry Dog.bmp	Prairie Wind.bmp	TEMP	lanma256.bmp	
Geometrix.bmp	Profiles	Upstream 16.bmp	lanmannt.bmp	
Gone Fishing.bmp	REGEDIT.EXE	WIN.INI	network.wri	

mkdir

mkdir - 지정된 등록부를 창조한다.

일반적으로 사용되는 선택항목은 다음과 같다.

-p　완전경로를 얻기 위하여 중간등록부를 창조한다. 아래로 여러개의 등록부층을 창조하려면 –p 를 리용하여야 한다.

mv

mv - 파일과 등록부의 이름을 변경한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-i　이미 존재하는 파일에 덧쓰기를 하겠는가를 확인하고 처리한다.

-f　파일이름이 다시 나타나면 작용하는 파일에 덧쓰기를 한다.

rm

rm - 파일과 등록부를 삭제한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-d　다른 파일형들과 같이 등록부를 삭제한다.

-i　사용자에게 이미 존재하는 파일을 삭제하지 않겠는가를 확인하고 삭제한다.

-f　파일들까지 삭제한다.

-r(-R)　등록부내용과 등록부이름자체를 반복하여 삭제한다.

touch

touch – 파일의 마지막처리시간 또는 파일창조시간 또는 수정에 대한 변경을 한다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-c　　파일이 존재하지 않으면 지정된 파일을 창조하지 않는다.

-f　　허락에 구애됨이 없이 파일을 변화시킨다.

다음의 실례는 touch에 의하여 file1을 창조하는 실례를 보여 준다.

D:\> **ls -1 filel**
ls:filel: No such file or directory

D:\> **touch filel**

D:\> **ls -1 filel**

-rw-rw-rw- 1 Administ Administ 0 Feb 20 11:45 filel

wc

wc – 단어, 행, 문자들의 개수를 나타낸다.

일반적으로 사용되는 선택항목들은 다음과 같다.

-l　　파일에서 행의 개수를 표시한다.

-w　　파일에서 단어의 개수를 표시한다.

-c　　파일에서 문자의 개수를 표시한다.

첫번째 실례에서 지령 wc를 리용하여 등록부의 내용을 표시하고 출력을 보여 준다. 두번째 실례에서 파일 system.ini에 대하여 wc통보를 만든다.

D:\> **1s**
CAT.EXE
CHMOD.EXE
CHOWN.EXE
CP.EXE
FIND.EXE

```
GREP.EXE
LN.EXE
LS.EXE
MKDIR.EXE
MV.EXE
RM.EXE
RMDIR.EXE
SH.EXE
TOUCH.EXE
VI.EXE
WC.EXE
D:\> ls | wc -wlc
    16      16      132
D:\> wc -wlc system.ini
    13      17      219     system.ini
```

보충적인 지령

Windows에는 지령행에 의하여 보여 줄수 있는 많은 보충적인 지령들이 있다. 여기서는 이 지령들의 일부를 보여 준다.

망지령

사용자는 Windows에서 지령행을 리용하여 많은 망지령들을 사용할수 있다. 이 항목에서는 아래에서 보여 준 일부 지령들에 대해서는 서술하지 않는다.

- **lpr**
- **route**
- **finger**
- **rexec**
- **ftp**
- **telnet**
- **hostname**
- **lpq**
- **tracert**
- **rcp**
- **rsh**
- **tftp**

telnet/?와 같이 입력재촉문에서 지령이름과 /?를 입구시키면 이 지령들에 대하여 더 많은것을 알수 있다. 아래에서는 자주 리용하는 보충적인 지령들을 보여 준다.

arp

arp 는 arp (Address Resolution Protocol) 고속완충기를 표시하고 편집하는데 사용되는데 이것은 물리적장치주소를 위한 IP 주소를 만든다. 고속완충기는 최근에 접근된 체계의 주소를 하나 또는 그이상 가지고 있다. 다음의 실례는 그것들의 물리적장치주소를 보여 주고 111 에서 최근에 처리된 체계주소와 주소 113 에서의 windows 체계에서 arp 지령을 리용하는것을 보여 준다.

d: **arp -a**

Interface: 159.260.112.113 on Interface 2

Internet Address	Physical Address	Type
159.260.112.111	08-00-09-f0-bc-40	dynamic

아래에는 arp/?지령을 리용하여 사용자가 볼수 있는 arp 지령의 몇가지 선택항목들이 제시되여 있다.

ipconfig

ipconfig 는 현재망대면부파라메터를 표시한다.

다음의 실례는 선택항목 /all 을 설정하여 주소 113 에서의 windows 체계에서 ipconfig 지령을 리용하는것을 보여 준다. 이것은 망대면부와 관계되는 모든 정보를 보여 준다.

d: **ipconfig /all**

Windows IP Configuration

Host Name ·················· : hpsysteml
DNS Servers ················· :
Node Type··················· : Broadcast
NetBIOS Scope ID············ :
IP Routing Enabled ··········· :No
WINS Proxy Enabled·········· :No
NetBIOS Resolution Uses DNS :No

Ethernet adapter Hpddnd3l:

Description··················· : HP DeskDirect
10/100 LAN Adapter
Physical Address ············· : 08-00-09-D9-9A-8A

DHCP Enabled · · · · · · · · · · · · · · : No
IP Address · · · · · · · · · · · · · · · · · · : 159.260.112.113
Subnet Mask · · · · · · · · · · · · · · · · : 255.255.255.0
Default Gateway · · · · · · · · · · · · · · : 159.260. 112. 250

ipconfig/지령을 리용하여 사용자가 볼수 있는 ipconfig 지령에 대한 몇가지 선택항목이 있다.

netstat

netstat 는 망규약의 통계를 제시하여 준다. 다음의 netstat 실례에서 선택항목 -e 와 -s 를 리용하는데 이것은 각각 Ethernet 통계와 여러가지 규약에 대한 통계를 보여 준다. 선택항목 -e 와 련관이 있는 Ethernet 통계의 아래에 "Interface 통계"가 주어 져 있고 그아래에 "IP statistics"가 주어 져 있다.

d: **netstat -e -s**

Interface Statistics

	Received	Sent
Bytes	3182007276	2446436
Unicast packets	11046	9604
Non-unicast packets	21827982	7932
Discards	0	0
Errors	0	1
Unknown protocols	4946670	

IP Statistics

Packets Received	=	20489869
Received Header Errors	=	133441
Received Address Errors	=	28222
Datagrams Forwarded	=	0
Unknown Protocols Received	=	0
Received Packets Discarded	=	0
Received Packets Delivered	=	20328206
Output Requests	=	12004
Routing Discards	=	0
Discarded Output Packets	=	0
Output Packet No Route	=	0
Reassembly Required	=	0
Reassembly Successful	=	0
Reassembly Failures	=	0

Datagrams Successfully Fragmented	= 0
Datagrams Failing Fragmentation	= 0
Fragments Created	= 0

ICMP Statistics

	Received	Sent
Messages	3702	23
Errors	0	0
Destination Unreachable	4	5
Time Exceeded	0	0
Parameter Problems	0	0
Source Quenchs	0	0
Redirects	3680	0
Echos	5	13
Echo Replies	13	5
Timestamps	0	0
Timestamp Replies	0	0
Address Masks	0	0
Address Mask Replies	0	0

TCP Statistics

Active Opens	= 27
Passive Opens	= 8
Failed Connection Attempts	= 1
Reset Connections	= 15
Current Connections	= 2
Segments Received	= 1888
Segments Sent	= 1854
Segments Retransmitted	= 3

UDP Statistics

Datagrams Received	= 607489
No Ports	= 19718827
Receive Errors	= 0
Datagrams Sent	= 10124

netstat/?지령을 리용하여 사용자가 볼수 있는 netstat 지령에 대한 몇개 선택항목이 있다. 만일 통계에 대한 값이 변하는것을 보려면 때때로 통계가 표시되도록 설정할수 있다.

ping

ping 은 망우에서 호스트가 련결되였는가를 확증하기 위하여 사용된다. ping 은 사용자가 지적한 목적호스트에로 파케트를 귀환할데 대한 매 요청이 있는 경우에 사용된다. 사용자가 지적할수 있는 ping 지령에는 여러가지 선택항목들이 있다.

다음의 실례는 파케트를 보낼것을 요구하는 시간수를 지정하는 선택항목 -n 과 최대값 8192 가 사용된 파케트의 길이를 지정하는 선택항목 -l 을 사용한데 대하여 보여주고 있다.

d: **ping -n 9 -1 8192 system2**

Pinging system2 [159.260.112.111] with 8192 bytes of data:

Reply from 159.260.112.111: bytes=8192 time=20ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=20ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=21ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=20ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=10ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=30ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=30ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=20ms TTL=255
Reply from 159.260.112.111: bytes=8192 time=20ms TTL=255

ping/?지령을 리용하면 사용자가 볼수 있는 ping 지령에 대한 몇가지 선택항목을 알수 있게 된다.

cacls 에 의한 허락

cacls 로 지령행에서 파일의 허락을 보거나 변경시킬수 있다.

그림 30-6 에서는 cacls 지령에 대한 도움말화면을 보여 준다.

cacls 는 파일들의 처리조종목록을 표시하고 수정하는데 사용된다. 그림 30-6 에서는 더 쉽고 상세하게 설명된 오른쪽 4 가지 처리형태에 대하여 볼수 있다. 다음목록은 cacls 지령과 련관된 처리의 오른쪽 부분을 생략한것에 대하여 보여 주고 있다.

N None
R Read
C Change
F Full Control

```
D:\>help cacls
Displays or modifies access control lists (ACLs) of files

CACLS filename [/T] [/E] [/C] [/G user:perm] [/R user [...]]
               [/P user:perm [...]] [/D user [...]]
   filename      Displays ACLs.
   /T            Changes ACLs of specified files in
                 the current directory and all subdirectories.
   /E            Edit ACL instead of replacing it.
   /C            Continue on access denied errors.
   /G user:perm  Grant specified user access rights.
                 Perm can be: R  Read
                              C  Change (write)
                              F  Full control
   /R user       Revoke specified user's access rights (only valid with /E).
   /P user:perm  Replace specified user's access rights.
                 Perm can be: N  None
                              R  Read
                              C  Change (write)
                              F  Full control
   /D user       Deny specified user access.
Wildcards can be used to specify more that one file in a command.
You can specify more than one user in a command.

D:\>
```

그림 30-6. cacls 도움말화면

그림 30-7. D:￥WINNT￥REGEDIT.EXE 허락

그림 30-7은 Cacls와 D:￥WINNT￥REGEDIT.EXE에 대한 허락을 보기 위한 파일 허락창문을 사용한것을 보여 준다. 이 파일은 체계에서 등록정보를 수정하거나 보기 위

하여 제시된 앞장들의 관점에서 보아도 가장 중요한 파일들중의 하나로 된다. 이 파일은 조작체계의 혼란을 피하기 위한 처리에서 오른쪽부분을 관리하기 위한 파일이다.

봉사기조작자들에게 제시되는 경고는 이 파일에 대하여 반드시 변경하게 한다. 사용자는 물론 이 경고에 대한 허락이나 체계상의 어떤 파일을 수정할수 있다. cacls 지령에 대한 파일허락창문과 출구는 여러가지 형식들로 할수 있으나 이 형식들은 같은 통보를 포함한다.

지령행 Backup

여벌복사(backup)를 초기화하기 위한 도형도구를 사용하는 선택점에서 지령행을 사용할수 있다. NTBACKUP 지령을 리용하여 사용자는 지령행에서 backup 을 초기화할수 있거나 이 지령을 사용하여 파일묶음을 만들수 있다. 사용자는 여벌복사를 위한 AT 지령과 함께 NTBACKUP 을 결합시킬수 있다. SCHEDULE 봉사를 시작한 사용자는 과제가 예정봉사되는 시간을 기입할수 있다. Backup 이 어떻게 결합되는가를 보기 위하여 다음의 2 개 지령을 고찰하자.

NTBACKUP

NTBACKUP−이 지령은 지령행에서 backup 을 초기화하기 위하여 사용된다.

NTBACKUP 지령의 형식 :

C:**NTBACKUP**/?

NTBACKUP operation path [/a] [/v] [/r] [/d "text"] [/b] [/hc:{on｜off}
[/t{option}] [/l "filename"] [/e] [/TAPE:{n}]

일반적으로 사용되는 선택항목들은 다음과 같다.

operation	backup 과 같이 실행하기 위한 조작이다.
path	사용자가 backup 을 요구하는 등록부이다.
/a	현재 일감(job)으로서 테프의 내용을 교체한다. 이 테프들에 backup 을 첨가한다.
/b	backup 에 국부등록부를 포함한다.
/d"text"	인용부호안에 표시된 text 에 의해 backup 모임의 내용이 설명된다.
/e	backup 가입은 완전 backup 가입을 포함하지 않는다.
/hc:｛on｜off｝	backup 를 위한 장치결합을 사용할것인가 사용하지 않을것인가를 지적한다. 사용자가 선택항목 /a 를 사용하지 않는다면 오직 이 선택항목만 사용할수 있다.
/L"filename"	backup 를 가입하기 위하여 사용하는 파일이름이다.
/r	tape 가 사용되는데 대해 제한된 처리를 한다. 사용자가 선택항목 /A 를 사용하지 않는다면 이 선택항목만 사용할수 있다.

/t option　normal, copy, incremental, differential, daily 와 같은 backup 의 형을 지적한다.
/tape:｛n｝　봉사기가 한개이상의 테프구동기를 가지고 있다면 사용될 테프구동기를 지적한다.
/v　조작을 검열한다.

일지파일에서 제외되는것만을 기록하면서 normal backup 으로 oracle 등록부를 여벌복사하기 위하여 테프를 확인하면서 "oracle backup"의 설명문을 포함하는 지령을 다음과 같이　쓸수 있다.

C:\NTBACKUP backup d:/oracle /e /v /t normal /d "oracle backup"

AT

AT – 이 지령은 일감(job)을 계획하는데 사용된다.

AT 지령의 형식 :

C:\ **AT**/?
AT [//computername] [id] [/DELETE] ｜/DELETE [/YES]]
AT [//computername] time [/INTERACTIVE] [/EVERY:date[, …] | [/NEXT: date [, …]] "command"

일반적으로 사용되는 선택항목들은 다음과 같다.

//computername　지령이 실행되는 콤퓨터이다.
id　계획된 지령을 식별하는 번호이다.
/DELETE　계획된 지령을 취소한다. 사용자는 id 또는 계획된 지령과 련관된 일감들을 취소하는데 id 를 사용할수 있다.
/YES　취소된 일감들을 확증할 때 YES 를 누른다.
time　계획되는　일감을 24h 형식으로 초기화한다.
/INTERACTIVE　일감이 배경에서 실행되게　된다.
/EVERY:date［, …］
요일들(월요일, 화요일 등과 같이) 또는 매달의 날자(1-31)로 계획된 일감들을 반복한다.
/NEXT:date［, …］이 선택항목은 다음에 일어 날 계획을 지적하는데 사용된다. 여러개 요일들(월요일, 화요일 등과 같이)과 달의 날자(1-31)를 지정한다.
"command "　실행될 지령이다.

계획된 backup 을 실행하기 위하여 앞의 NTBACKUP 지령과　AT 를 결합할수 있다. 이전의 NTBACKUP 지령들을 결합하여 다음지령을 얻을수 있다.

C:\ AT 03:00 /every:Monday,　Tuesday,　Wednesday,　Thursday,　Friday "backup"

제31장. UNIX를 위한 봉사(SFU)

SFU에 대한 개괄

SFU (Microsoft Services for UNIX)는 많은 본질적인 측면에서 UNIX와 Windows 사이의 **호상조작성**(Interoperability)을 준다. Microsoft는 SFU에서 호상조작성으로 널리 사용되는 세개 부분처리묶음을 가지고 있다. 봉사프로그람 NFS(Network File System), Telnet, UNIX와 같은 중요한 UNIX와 Windows 호상조작성함수들은 SFU의 부분이다. 그림 31-1은 SFU를 Windows 체계에 적재할 때 표시되는 차림표를 따내는것과 같은 기능을 보여 준다. 이 장에서는 NFS로부터 시작하여 순차적으로 SFU에 대한 가장 중요한 기능적측면들을 고찰한다.

SFU의 NFS기능을 사용

사용자가 SFU의 NFS 기능으로 자기의 Windows 체계에 UNIX의 파일체계를 설치하기 위해서는 전형적인 NFS의뢰기를 시동시켜야 한다.

SFU의 NFS 의뢰기는 대리인으로 작용함으로써 사용자의 Microsoft Server Message Block(SMB)망을 련결한다. 앞으로 SMB는 Windows의뢰기로 UNIX NFS봉사를 요구한다.

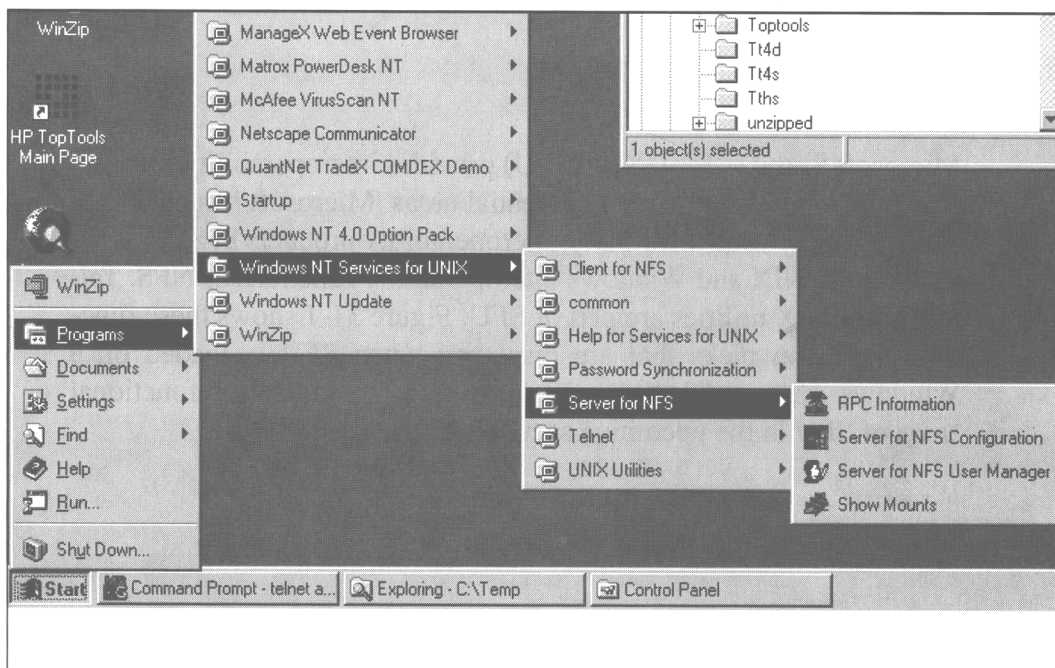



그림 31-1. SFU의 차림표구조

Windows 에서는 NFS 를 실행하는 의뢰기측면에 언제나 주의를 돌리게 된다. 왜냐하면 UNIX 체계가 보통 Windows 사용자들이 접근하기 위한 더 크고 중요한 체계이기때문이다. 따라서 Windows 사용자들은 일반적으로 Windows 체계우에 UNIX 등록부를 태운다.

이것은 반드시 필요하지는 않다. 사용자는 NFS 봉사기와 같이 SFU 로 Windows 체계를 설치할수 있다. 그림 31-1 은 체계를 설치한 후 SFU 의 차림표구조를 보여 준다. Server for NFS 차림표가 선택되였다.

작업을 시작하기 위하여서는 선택항목 Client for NFS 를 지적한다. 왜냐하면 NFS 의뢰기가 Windows 와 UNIX 가 혼잡된 환경에서 광범히 사용되며 후에 NFS 봉사기에로 쉽게 돌아 올수 있기때문이다.

그림 31-2. SFU 인증

Client for NFS Configuration 에 NFS 와 관련된 다음의 6 가지 종류의 기능이 있다.

Authentication
Mount Options
File Access
Filenames
Configured NFS LANs
Symbolic Links

인증(Authentication)으로 시작하는 매 과정을 그림 31-2 에서 보여 주고 있다.

인증은 NFS 봉사기에 접속하기 위한 몇가지 기초통보를 추가할것을 요구한다. 그림에서 User Name 과 Password 는 NFS 봉사기에 접속하기 위하여 사용된다. 이것들은 봉사기에 접속을 하기 위한 절차로서 봉사기에 설치되여야 한다. 사용자는 이 실례의 부분인 NIS 도 선택할수 있다. NFS 를 접속하기 위한 봉사기는 PC NFS 기법을 실행하여야 한다. 봉사기는 이 기법을 실행하겠는가 혹은 안하겠는가를 선택할수 있다.

그림 31-3 은 NFS 봉사기에서 pcnfsd 를 선택하는것을 보여 준다.

```
Command Prompt - telnet
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
# ps -ef | grep nfs
    root 11935 11918  2 22:13:55 ttyp1     0:00 grep nfs
    root  1255     1  0  Feb  1  ?         0:00 /usr/sbin/nfsd 4
    root  1256  1255  0  Feb  1  ?         0:00 /usr/sbin/nfsd 4
    root  1258  1255  0  Feb  1  ?         0:00 /usr/sbin/nfsd 4
    root  1257  1255  0  Feb  1  ?         0:00 /usr/sbin/nfsd 4
    root  7489     1  0 07:09:55 ?         0:00 /usr/sbin/rpc.pcnfsd
#
```

그림 31-3. UNIX 체계에서 pcnfsd 의 검사

ps 의 마지막부분에서는 pcnfsd 가 NFS 봉사기에서 실행된다는것을 보여 준다.

지적된 User Name, Password 와 PCNFSD 봉사기를 NFS LAN 구성(Configured NFS LAN)에 이동시킬수 있다.

그림 31-4 는 NFS 를 사용할것을 요구할 때 두개의 LAN 을 구성하는것을 보여 준다.

그림 31-4. NFS LAN을 구성하는 SFU

그림 31-5. 구체적인 LAN 편집

Edit …를 선택하면 그림 31-5에서 Broadcast Address와 같은 통보를 볼수 있다.

다음으로 사용자가 NFS와 접속을 할 때 조종해야 하는 기호적련결방법을 선택한다. 기호적련결은 이미 존재하는 파일 혹은 등록부에서의 파일이나 등록부에 대한 **사영**(Mapping)이다. Resolve Symbolic Link 를 선택하려면 련결을 포함하는 현재 경로이름을 보아야 한다.

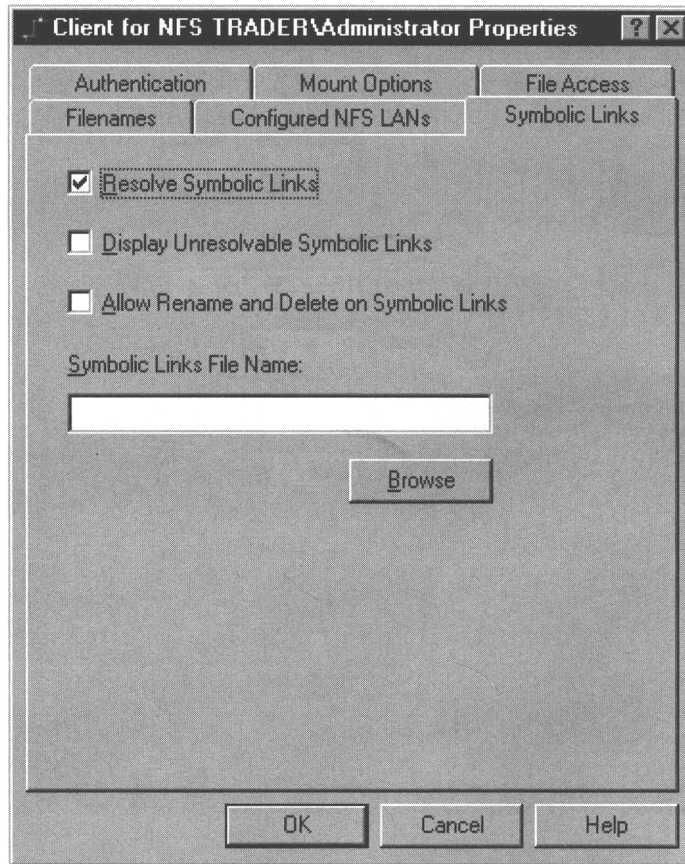



그림 31-6. NFS를 위한 의뢰기 SFU

Symbolic Link에 대한 해당한 선택항목들은 그림 31-6에 제시되였다.

전형적인것들은 기호적련결을 풀수 있으나 련결을 풀수 없는것들도 있고 표시할수 없는것들도 있다. Windows 와 UNIX 에는 파일이름을 짓는데 리용되는 일부 각이한 관례들이 있다.

SFU 는 그림 31-7 에서 보여 준것과 같은 몇가지 파일이름과 관계되는 선택항목들을 준다.

모든 새로운 파일이름들은 소문자로 쓴다. 이것은 여러개 조작체계로 작업할 때 혼돈을 적게 하기 위해서이다. 이미 있는 파일이름으로 작업해도 좋고 그림 31-7 에서와 같이 파일이름들을 선택하고 선택된 파일을 지적하여 작업할수도 있다.

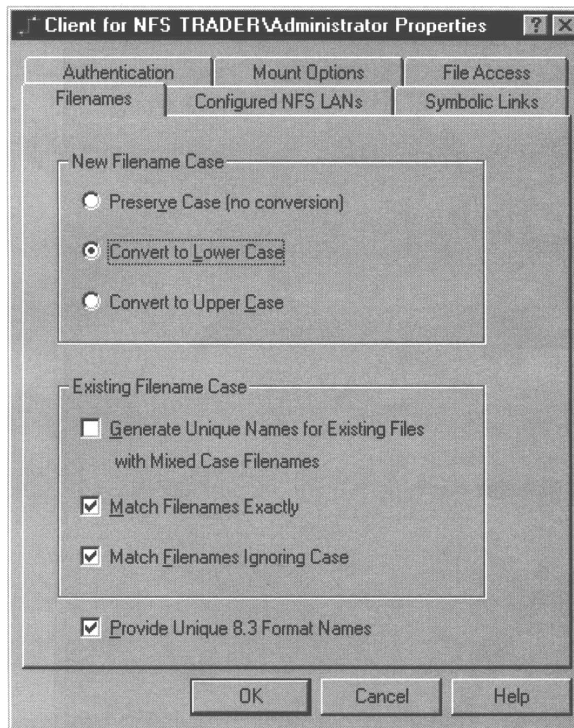



그림 31-7. NFS를 위한 의뢰기 SFU

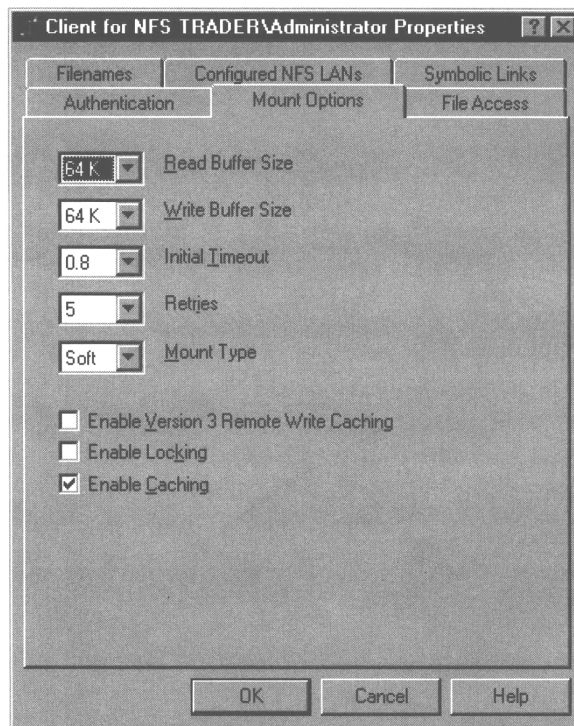



그림 31-8. 선택항목들을 설치하는 SFU

NFS 에서 작업할 때 태워 놓은 선택들이 많을수 있다. 그림 31-8 은 선택항목들을 태우는것을 보여 준다.

Read Buffer Size 는 기정값으로 64000byte 를 사용한다. 이 수는 크게 설정하는것이 좋다. 왜냐하면 묶음자료부분이 NFS 에 대하여 읽을 동안 사용되기때문이다. 일반적으로 큰것이 좋다. Write Buffer Size 도 마찬가지이다. Initial_Timeout 는 고치기전까지 봉사기로 응답하는 시간의 총합으로 지정된다. Retries 는 조작하기전에 봉사기에 접근한 시간수자를 지정한다. Mount Type 는 Soft 를 지적하고 Enable Caching 을 선택한다.

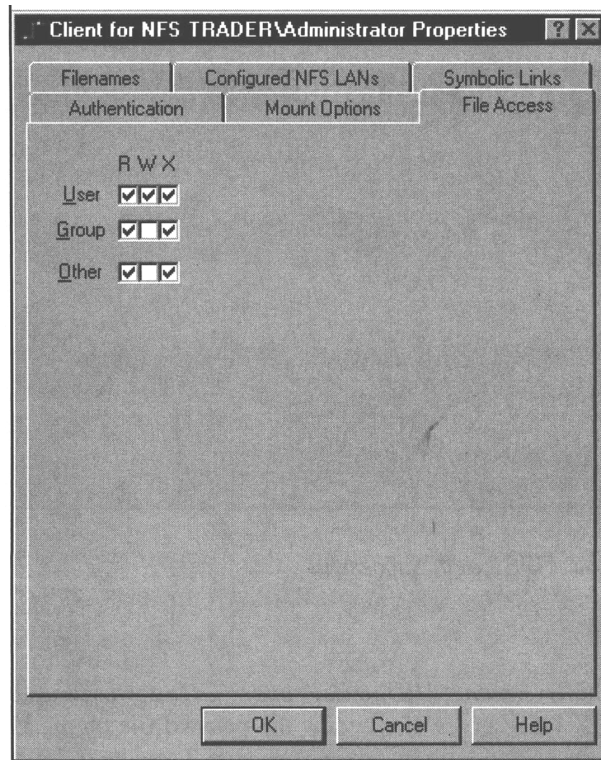



그림 31-9. 파일접근 SFU

File Access 는 사용자가 NFS 를 태우는것을 설정할 권한이 있다는것을 지정한다. 파일은 그림 31-9 에서 습관적으로 선택하는 우선권기능을 보여 준다.

암시적으로는 사용자가 파일에로의 접근을 제한하지 않는데 Group 과 Other 에 있는 것들은 읽기(R)와 실행(X)만 처리한다. 이것들은 공통적인 우선권기능이다.

그림 31-2 에서 보여 준 인증창문에서 OK 를 선택하면 NFS 련결을 쉽게 할수 있다. 이때 그림 31-10 에서 보여 준것과 같은 사용자이름, 다른 정보를 확인하는 창이 나타난다.

Username, UID, Primary GID 는 UNIX NFS 봉사기에 대응한것이다. UNIX NFS 봉사체계에서 passwd 파일을 보고 사용자 hp 를 찾는다면 다음의 내용을 볼수 있다.

hp:EkyXw/N. EwFNw:104:20::/home/hp:/sbin/sh

이 passwd 입장에서 통보는 NFS Login 창문에서 보여 준것과 일치한다. login 를 처리한후 Windows 체계에서 Explorer 를 사용하면 UNIX NFS 봉사기에서 보낸 파일체계를 볼수 있다.

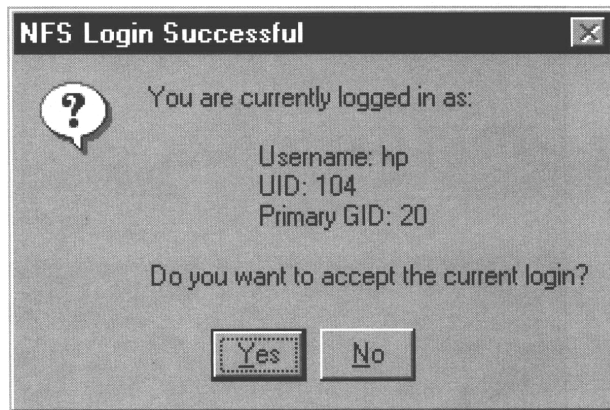



그림 31-10. NFS 가입성공

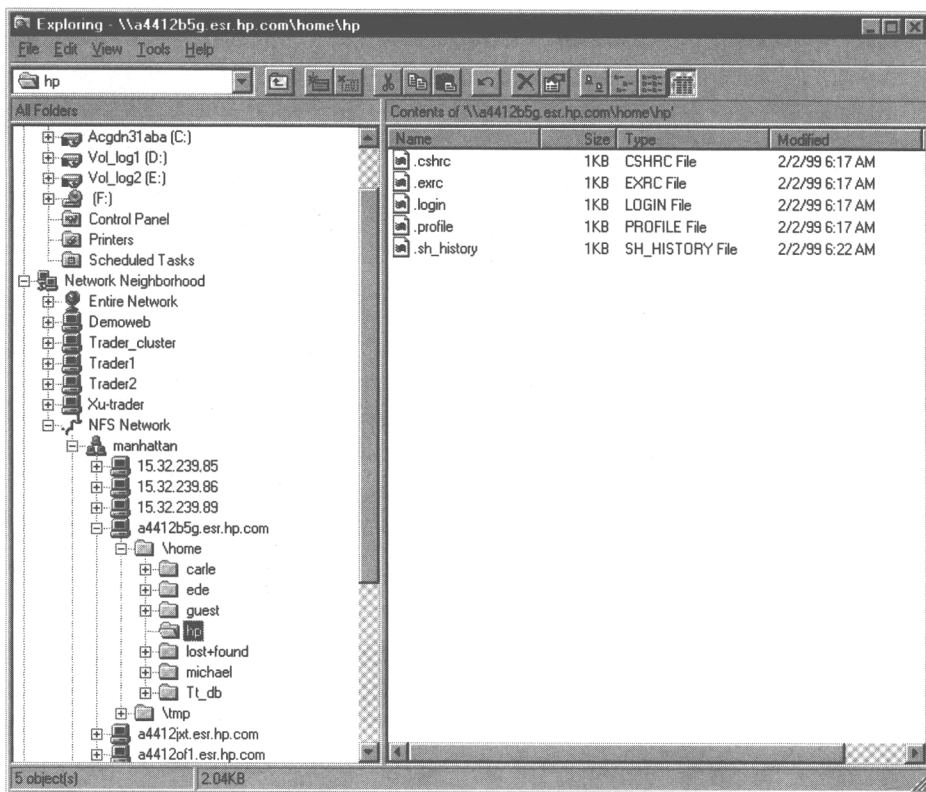



그림 31-11. 파일체계를 설치한 NFS 보기

그림 31-11 은 선택된 구체적인 체계로서 manhattan LAN 을 보여 준다.

그림에서 \home\hp 가 선택되였을 때 Explorer 창문의 오른쪽에서는 \home\hp 등록부의 파일들을 보여 준다. hp 등록부에는 파일들을 다루기 위한 허락이 있다.

.cshrc 와 같은 파일들을 선택하면 이 파일을 검열할수 있고 그 파일의 속성을 볼수 있다. 그림 31-12 는 속성창문을 보여 준다.

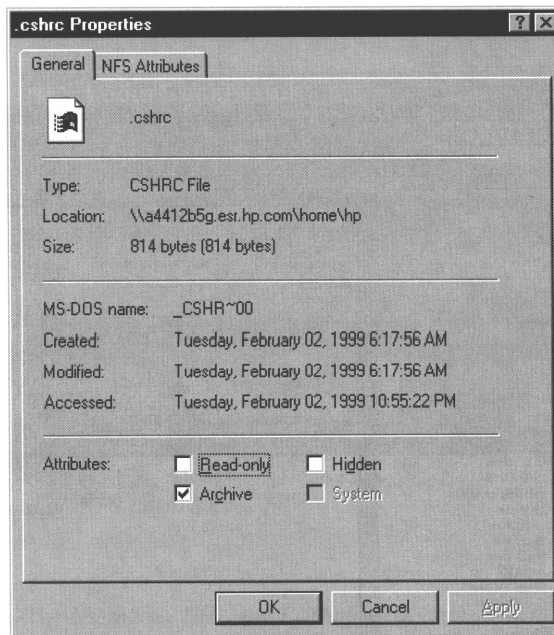



그림 31-12. SFU 속성

이 창문에서는 Read-only 가 선택되여 있지 않다. 그러므로 이 파일에 대한 완전한 처리를 할수 있다.

Telnet 의뢰기

SFU 의 부분으로서 Telnet 의뢰기도 있다. SFU 차림표에서 Telnet 의뢰기를 선택하면 그림 31-13 에서 보여 준 창문이 나타난다.

이 Telnet 창문에서는 \home\hp 의 내용에 대한 긴 목록을 조성하였다.

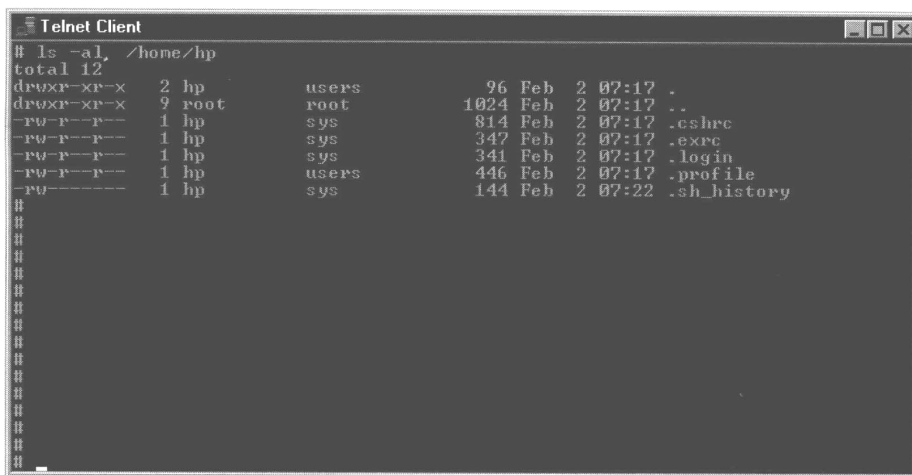
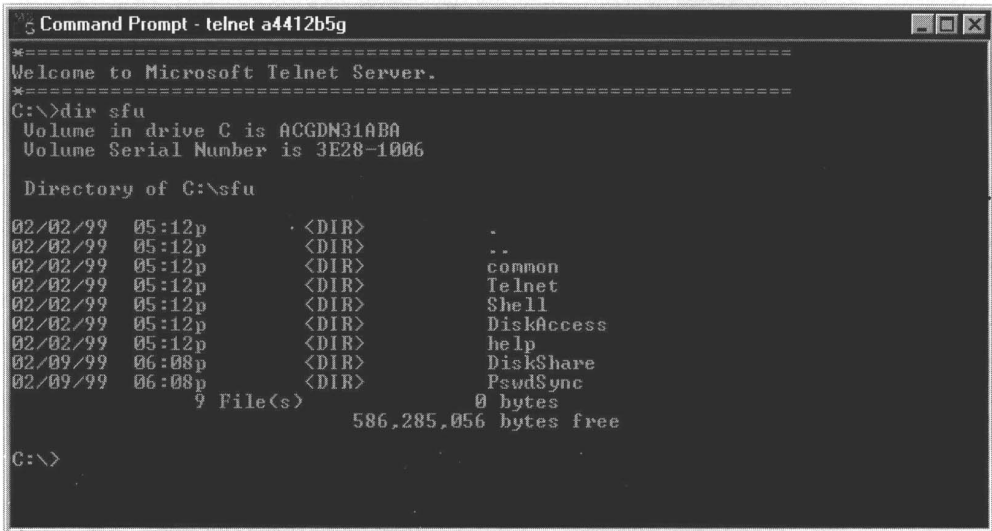



그림 31-13. Telnet 의뢰기의 SFU

Telnet 봉사기

SFU 의 부분으로서 적재된 Telnet 봉사기도 있다. 이것은 사용자가 UNIX 체계 또는 Telnet 의뢰기와 함께 다른 임의의 체계로부터 Telnet 봉사기와 함께 Windows NT 체계에 련결할수 있다는것을 의미한다. 그림 31-14 는 UNIX 체계에서 Windows Telnet 봉사기를 처리하는 과정을 보여 준다.

```
Command Prompt - telnet a4412b5g
*===============================================================
Welcome to Microsoft Telnet Server.
*===============================================================
C:\>dir sfu
 Volume in drive C is ACGDN31ABA
 Volume Serial Number is 3E28-1006

 Directory of C:\sfu

02/02/99  05:12p        <DIR>          .
02/02/99  05:12p        <DIR>          ..
02/02/99  05:12p        <DIR>          common
02/02/99  05:12p        <DIR>          Telnet
02/02/99  05:12p        <DIR>          Shell
02/02/99  05:12p        <DIR>          DiskAccess
02/02/99  05:12p        <DIR>          help
02/09/99  06:08p        <DIR>          DiskShare
02/09/99  06:08p        <DIR>          PswdSync
               9 File(s)              0 bytes
                          586,285,056 bytes free

C:\>
```

그림 31-14. Telnet 봉사기의 SFU

그림 31-14 에서는 UNIX 체계에서 Windows 체계에로 Telnet 시작을 초기화한다. Microsoft Telnet 봉사기로부터 Welcome 정보를 받은후 사용자는 화면에서 제시된 dir 와 같은 지령을 사용할수 있다. 이것은 Windows 체계에서 작업할 때 지령을 주는것과 같다.

SFU Telnet 봉사기기능은 UNIX 체계로부터 Windows 체계에로의 일부 직접처리를 할수 있다. 이것은 혼잡된 환경에서 큰 도움을 주는것으로 된다. 이 기능은 이 Telnet 련결 또는 Windows 체계에서 직접 실행할수 있는 많은 UNIX 봉사프로그람들에 의해 더 한층 증가된다. 다음부분에서 "UNIX 응용프로그람"이라는 봉사프로그람을 취급한다.

UNIX 편의프로그람

SFU 의 한가지 좋은 기능은 Unix Utilities-Unix Command Shell 로 불러 내는 UNIX 지령표시이다. 그림 31-15 는 UNIX 편의프로그람의 목록으로 열려 진 창문을 보여 준다.

```
Unix Command Shell
$ ls
basename.exe   dirname.exe   ln.exe      rm.exe      tail.exe
cat.exe        egrep.exe     ls.exe      rmdir.exe   tee.exe
chmod.exe      fgrep.exe     mkdir.exe   sed.exe     touch.exe
chown.exe      find.exe      more.exe    sh.exe      uniq.exe
cp.exe         grep.exe      mv.exe      sh_histo    vi.exe
cppstdin.ksh   head.exe      perl.exe    sort.exe    wc.exe
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
```

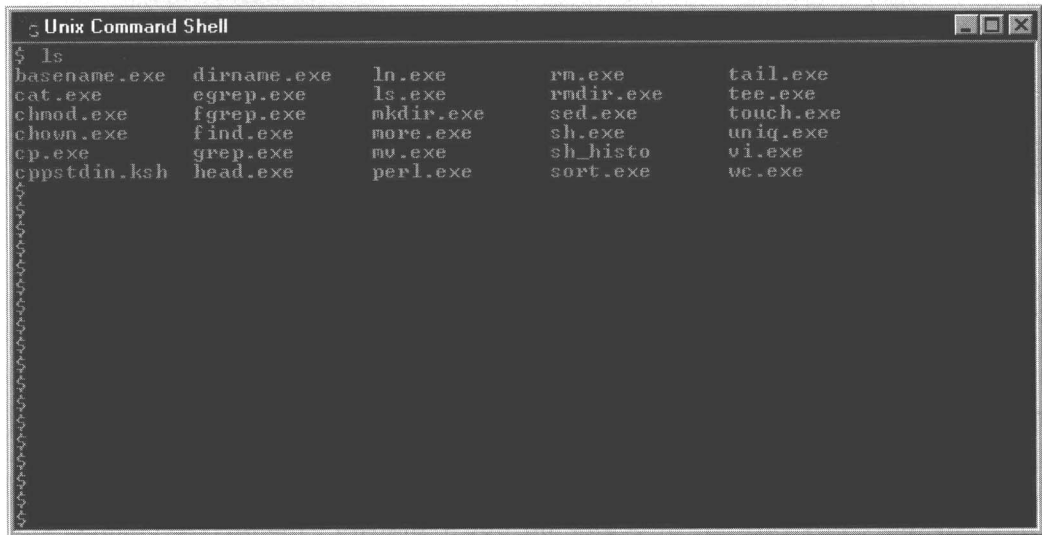



그림 31-15. SFU UNIX 편의프로그람목록

그림 31-15에서 제시된 편의프로그람들은 바로 그 편의프로그람들이 UNIX에서 작업한 방법으로 SFU에서도 작업한다. 사용자는 또 다른 체계로부터 만들수 있는 Telnet 시작을 통해 UNIX 편의프로그람들을 처리한다. 이 편의프로그람이 어떻게 작업하는가를 보여 주기 위하여 그림 31-16에서 보여 준것과 같은 여러 지령들을 사용한다.

```
Unix Command Shell
$ pwd
C:/SFU/Shell
$ cd /
$ pwd
C:/
$ ls
AUTOEXEC.ADT                               Multimedia Files
AUTOEXEC.BAT                               NT4SP3
Acrobat3                                   NTOPTION
BOOT.BAK                                   Program Files
COMMAND.COM                                SETUP
CONFIG.SYS                                 SFU
DMI                                        TT4D
FIRSTBOO.TXT                               TT4S
HP_INFO                                    TTHS
I386                                       Temp
IE401SP1                                   Toptools
Inetpub                                    WINNT
Internet Explorer 4.01 SP1 Setup           boot.ini
LAN                                        pagefile.sys
MOUSE                                      webhelp
MktData
$
$
$ _
```

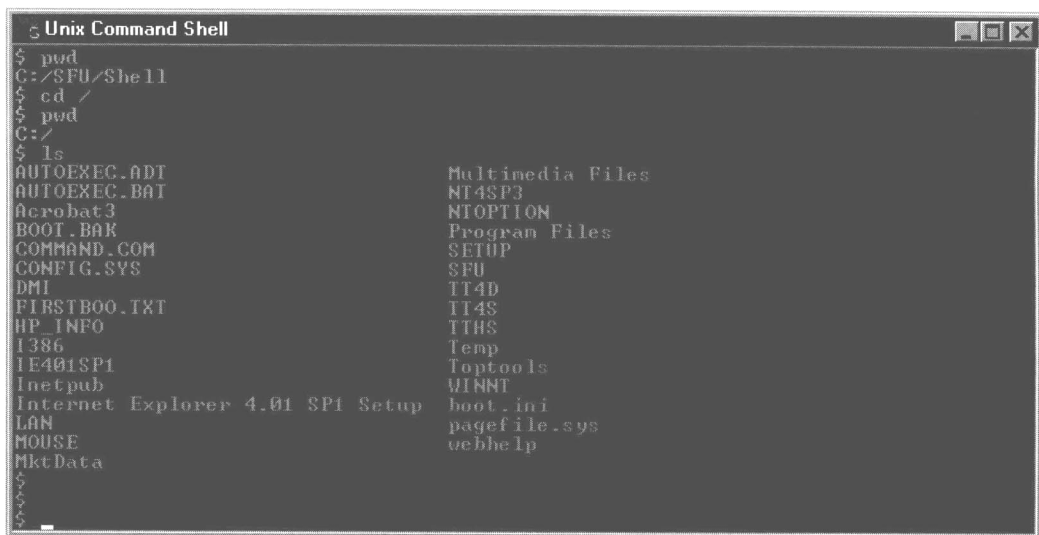



그림 31-16. 여러개 UNIX 편의프로그람을 리용하는 실례의 SFU

이 창문은 UNIX 편의프로그람을 요구할 때 Windows 체계상에서 C:/SFU/Shell 등록부에 있다는것을 보여 준다. UNIX 체계상에서 고찰하는것처럼 cd/를 사용하여 등록부를 C:로 변경시킬수 있다. pwd 는 C:에 있다는것을 확인한다. C:에 있는 파일들을 보기 위하여 ls를 사용한다.

그림 31-17 에서 보여 준바와 같이 이 파일들이 Windows 들에서 어떻게 실행되는가를 더 잘 알기 위하여 UNIX 편의프로그람을 2개이상 사용한다.

```
Unix Command Shell
$ ls | grep -i s
CONFIG.SYS
FIRSTBOO.TXT
IE401SP1
Internet Explorer 4.01 SP1 Setup
MOUSE
Multimedia Files
NT4SP3
Program Files
SETUP
SFU
TT4S
TTHS
Toptools
pagefile.sys
$ ls | grep -i s | wc
     14      20     166
$
$
$
$
$
$
$
$
$
$ _
```

그림 31-17. 여러개 UNIX 편의프로그람을 사용하는 실례의 SFU

이 창문에서는 -i 인 경우를 무시하고 s 를 찾는 /s 와 grep 지령을 사용한다. 이 창문에서는 대문자와 소문자 s 를 포함한 파일들을 보여 준다. 마지막으로 단어의 개수를 wc에 출구한다.

NFS 봉사기

Windows 체계는 NFS 의뢰기로 동작할수도 있고 SFU 와 함께 NFS 봉사기로도 동작할수 있다. NFS 를 실행하는 체계는 Windows 체계에 보낸 파일체계를 태울수 있다.

Server for NFS Configuration 에서 사용자는 NFS 봉사기설치의 모든 문제를 구성할수 있다. 구성의 대부분항목들은 암시적으로 설정하였는데 그것들이 여기서 실행된다. 사용자는 NFS 봉사를 진행하기 위하여 후에 보충적인 구성을 실행할것을 요구할수 있다.

실례로 그림 31-18 에서는 한개 Share Name 을 창조한것을 보여 준다.

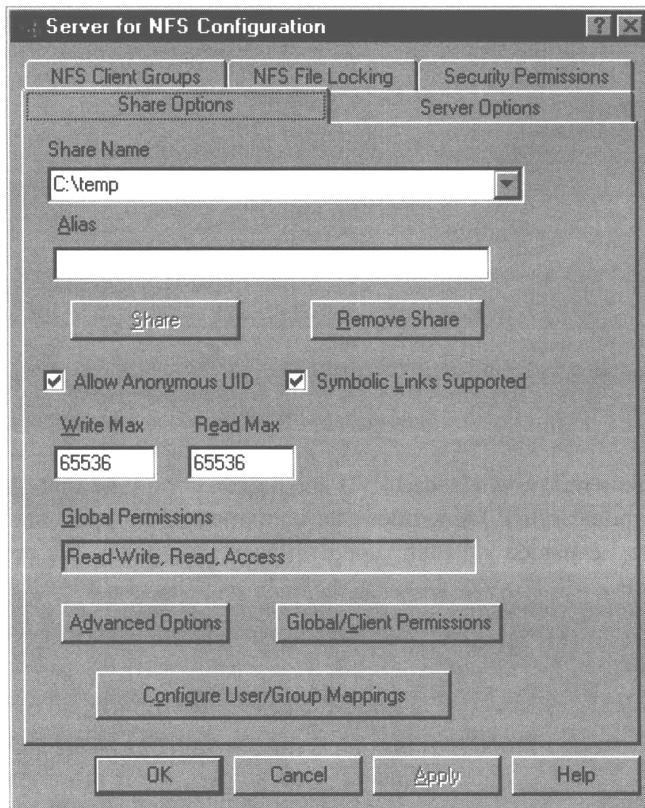



그림 31-18. 공유이름의 SFU

차림표의 Share Option 에서 Server for NFS Configuration 을 선택함으로써 그림 31-18 에서 보여 준 Share Name C:\temp 에 가입한다. 그림 31-19 에서는 보내진 파일이 무슨 파일체계인가를 보기 위한 Mount Information 을 보여 준다.

C:\temp 의 파일체계는 아무런 제한없이 보낸 목록에 있다. 설정된 Share Name 으로 모든 여러가지 NFS 봉사구성을 위한 기정적인것을 사용할수 있으며 UNIX 체계상에 C:\temp 를 태울수 있다. C:\temp 를 설치하기 위하여 사용자는 UNIX 체계에서 다음지령을 사용한다.

```
#mount 19.32.23.112:C:\temp /ntmount
```

이 지령을 리용하면 대다수의 UNIX 체계상에서 작업할수 있다. 태우는 지령에서 우선 사용자가 root 에서처럼 규정대로 사용하는 설치지령을 사용하였다. 우의 지령에서 그 다음에 쓴것은 그우에 설치할것을 요구하는 파일체계를 가진 Windows 체계의 이름이다. 이 경우에 체계이름대신에 IP 주소를 사용한다. 체계이름 또는 IP 주소는 두점(:)뒤에 온다. 우의 지령에서 그 다음에 쓴것은 C:\temp 인 경우에 그 파일체계가 Windows 체계상에 나타날 때 태우려고 하는 파일체계이름이다. 마지막에 쓴 /ntmount 은 C:\temp 를 태울 UNIX 체계상의 등록부이름이다. mount 지령으로 여러가지 각이한 선택을 추가할수 있는데 실례에서는 모든 기정적인 설정을 사용하였다.

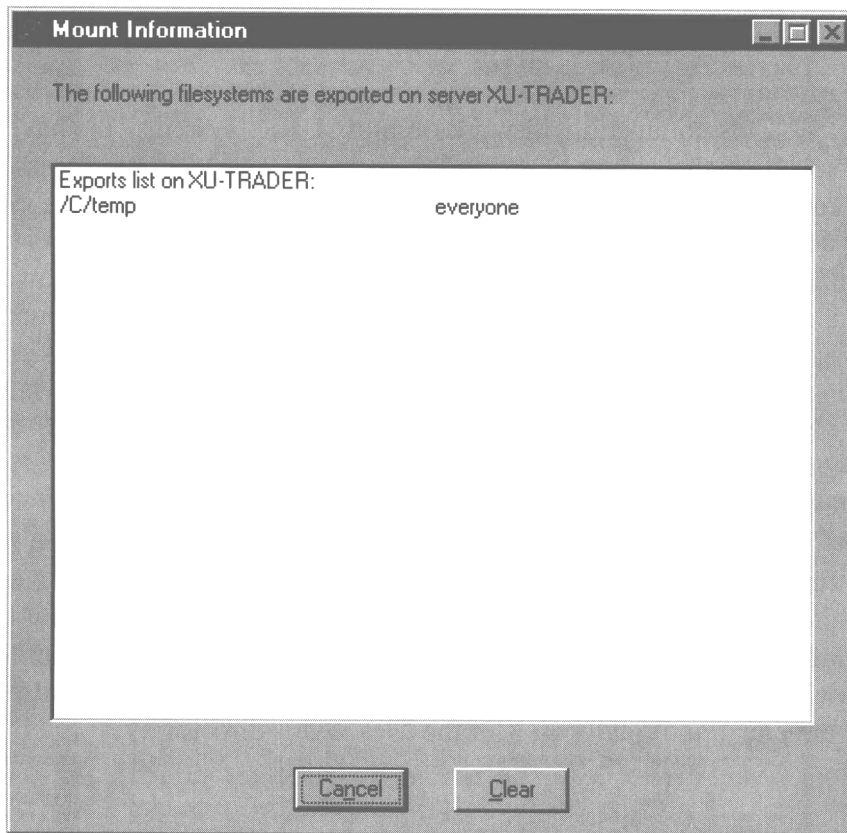



그림 31-19. 통보설치 SFU

여기서 지정한 NFS 설치가 UNIX 체계에 설정되였는가를 보기 위하여 검사하여 보자. 다음실례는 설치가 설정되였는가를 보기 위한 UNIX 체계에서 bdf 지령을 사용한것을 보여 준다(이것은 이 책의 앞에서 취급한 df 지령과 어느 정도 류사하다.). 아래에서 등록부 ntmount에 대한 cd 지령과 파일에 대한 ls를 보여 준다.

bdf

Filesystem	kbytes	used	avail	%used	Mounted on
/dev/vg00/lvol3	151552	53165	92162	37%	/
/dev/vg00/lvol1	47831	14324	28722	33%	/stand
/dev/vg00/lvol8	163840	87595	71039	55%	/var
/dev/vg00/lvol7	339968	314984	23189	93%	/usr
/dev/vg00/lvol6	102400	62284	37607	62%	/tmp
/dev/vg00/lvol5	1048576	656649	367439	64%	/opt
/dev/vg00/1vo14	69632	32448	34850	48%	/home
/dev/vgCE/1vpatch	1024000	1357	958732	0%	/ce/patches
/dev/vgCE/lvfw	512000	1314	478856	0%	/ce/firmware
/dev/vgCE/cetmp	512000	419166	87028	83%	/ce/ce-tmp

```
19.32.23.112:temp      2096160      1523360    572800      73%          /ntmount
# cd /ntmount
# ls
_istmpo.dir         test.html          ~df8ec8. tmp        ~dfd65e. tmp
_istmp1.dir         tt2.exe            ~df8ee7. tmp        ~dfd65f. tmp
_istmp2.dir         tt3.exe            ~dfa393. tmp        ~dfd660. tmp
ie401spl.exe        ttdemo.zip         ~dfa394. tmp        ~dfd66b. tmp
ie4setup.exe        ttwiz(l).exe       ~dfa3a3. tmp        ~dfd66c. tmp
jack.log            wbemcore.exe       ~dfa3a4. tmp        ~dfe649. tmp
jack1.log           winzip70.exe       ~dfb2a.  tmp        ~dfe64a. tmp
jack2.1og           ~df7e2f.tmp        ~dfb39.  tmp        ~dfe64b. tmp
mmcl4.tmp           ~df7e40.tmp        ~dfb3a.  tmp        ~dfe64c. tmp
mmcaaa7.tmp         ~df7e4l.tmp        ~dfd63c. tmp        ~dfe659. tmp
mmcaaad.tmp         ~df7e42.tmp        ~dfd63d. tmp        ~dfe65a. tmp
mmcaabl.tmp         ~df7e4f.tmp        ~dfd64c. tmp        ~dfe65b. tmp
nph-ntfinal.exe     ~df7e50.tmp        ~dfd64d. tmp        ~dfe65c. tmp
ntagt33e.exe        ~df7e51.tmp        ~dfd64e. tmp        ~dfe65d. tmp
ntoption. exe       ~df7e52.tmp        ~dfd64f. tmp        ~dfe669. tmp
sfu                 ~df7e53.tmp        ~dfd65b. tmp
sful                ~df8e99.tmp        ~dfd65c. tmp
temp. log           ~df8ea9.tmp        ~dfd65d. tmp
```

이 지령을 수행한후에 UNIX 체계상에서 태워 진 하나의 파일체계인 /ntmount 를 볼수 있다. 이 시점에서 root 가 아닌 사용자도 변경할수 있다. 왜냐하면 일반적으로 파일체계를 태운 다음 NFS 에서 root 로 파일들을 다루는것이 적합치 않을수 있기때문이다. 여기서는 hp 사용자로 변경한다. 다음등록부를 /ntmount 로 변경하고 Windows 체계에서 C:\temp 에서 그것들과 일치한 내용을 보다 긴 목록으로 표시하여 hp 자격으로 파일들을 볼수 있다.

```
$ ll
total 5710
drwxrwxrwx    2   hp   users          64   Feb    9   17:58    _istmp0.dir
drwxrwxrwx    3   hp   users          96   Feb    2   10:18    _istmpl.dir
drwxrwxrwx    2   hp   users          64   Feb    2   10:19    _istmp2.dir
-rwxrwxrwx    1   hp   users    24227193   Feb    1   19:50    ie401spl.exe
-rwxrwxrwx    1   hp   users      443160   Jan   10   10:21    ie4setup.exe
-rwxrwxrwx    1   hp   users         218   Feb   10   1999     jack.log
-rwxrwxrwx    1   hp   users         218   Feb   10   1999     jack1.log
-rwxrwxrwx    1   hp   users         218   Feb   10   1999     jack2.log
-rwxrwxrwx    1   hp   users       60416   Feb    2   15:26    mmcl4.tmp
-rwxrwxrwx    1   hp   users      102400   Feb    9   09:24    mmcaaa7.tmp
```

-rwxrwxrwx	1	hp	users	102400	Feb	9	12:40	mmcaaad.tmp
-rwxrwxrwx	1	hp	users	102400	Feb	9	12:40	mmcaabl.tmp
-rwxrwxrwx	1	hp	users	464200	Feb	1	19:33	nph-ntfinal.exe
-rwxrwxrwx	1	hp	users	5514240	Feb	1	17:43	ntagt33e.exe
-rwxrwxrwx	1	hp	users	38940572	Feb	1	20:07	ntoption.exe
drwxrwxrwx	20	hp	users	640	Feb	3	09:58	sfu
drwxrwxrwx	5	hp	users	160	Feb	10	1999	sful
-rwxrwxrwx	1	hp	users	218	Feb	2	11:34	temp.log
-rwxrwxrwx	1	hp	users	35	Feb	1	19:48	test.html
-rwxrwxrwx	1	hp	users	7046431	Feb	1	20:19	tt2.exe
-rwxrwxrwx	1	hp	users	5758110	Feb	1	20:21	tt3.exe
-rwxrwxrwx	1	hp	users	3232301	Feb	3	15:18	ttdemo.zip
-rwxrwxrwx	1	hp	users	1086772	Feb	1	16:53	ttwiz(l).exe
-rwxrwxrwx	1	hp	users	3456925	Feb	1	19:55	wbemcore.exe
-rwxrwxrwx	1	hp	users	943949	Feb	3	15:51	winzip70.exe
-rwxrwxrwx	1	hp	users	4096	Feb	2	15:26	~df7e2f.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e40.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e4l.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e42.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	2	15:26	~df7e4f.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	2	15:26	~df7e50.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e5l.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e52.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	2	15:26	~df7e53.tmp
-rwxrwxrwx	1	hp	users	9728	Feb	9	09:24	~df8e99.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	9	09:24	~df8ea9.tmp
-rwxrwxrwx	1	hp	users	5120	Feb	9	09:24	~df8ec8.tmp
-rwxrwxrwx	1	hp	users	3584	Feb	9	09:24	~df8ee7.tmp
-rwxrwxrwx	1	hp	users	6144	Feb	2	15:26	~dfa393.tmp
-rwxrwxrwx	1	hp	users	9728	Feb	2	15:26	~dfa394.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	2	15:26	~dfa3a3.tmp
-rwxrwxrwx	1	hp	users	5120	Feb	2	15:26	~dfa3a4.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	9	09:24	~dfb2a.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	9	09:24	~dfb39.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	9	09:24	~dfb3a.tmp
-rwxrwxrwx	1	hp	users	4608	Feb	9	09:24	~dfd63c.tmp
-rwxrwxrwx	1	hp	users	16384	Feb	9	09:24	~dfd63d.tmp
-rwxrwxrwx	1	hp	users	4608	Feb	9	09:24	~dfd64c.tmp
-rwxrwxrwx	1	hp	users	3072	Feb	9	09:24	~dfd64d.tmp

```
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd64e.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd64f.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd65b.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd65c.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd65d.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd65e.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd65f.tmp
-rwxrwxrwx  1  hp  users  8192  Feb  9  09:24  ~dfd660.tmp
-rwxrwxrwx  1  hp  users  4608  Feb  9  09:24  ~dfd66b.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfd66c.tmp
-rwxrwxrwx  1  hp  users  4096  Feb  9  09:24  ~dfe649.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe64a.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe64b.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe64c.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe659.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfe65a.tmp
-rwxrwxrwx  1  hp  users  3072  Feb  9  09:24  ~dfe65b.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe65c.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe65d.tmp
-rwxrwxrwx  1  hp  users  3584  Feb  9  09:24  ~dfe669.tmp
```

우리는 사용자 hp 와 일치한 user 의 그룹이 이 긴 목록의 부분이라는것을 볼수 있다.

이 부분에서 실행한 NFS 봉사기설치는 Windows 파일체계가 NFS 봉사기에서처럼 보여 지며 NFS 의뢰기의 부분처럼 중요하게 되도록 하기 위하여 앞서 실행된 NFS 의뢰기와 함께 결합할수 있다. NFS 를 사용하는 범위는 사용자가 파일공유방법의 부분에서처럼 환경을 어떻게 설정하는가에 달려 있다. 왜냐하면 NFS 는 대부분 UNIX 방안에서 쓸수 있으며 사용자는 Windows 상에서 NFS 를 사용하여 환경설정을 할수 있기때문이다. 사용자가 Windows 와 UNIX 의 혼잡된 환경에서 힘든 NFS 을 사용하려 한다면 단순한것으로부터 시작해야 하며 꼭 필요한것들의 실행은 NFS 를 사용하여 공유된 열쇠등록부들에 의하여 검사할수 있다. 앞의 실례에서 본것처럼 Windows 체계에서 NFS 는 Windows 체계와 UNIX 체계의 혼잡된 환경에서 공유되여 있는 파일을 상당히 편리하게 한다.

통과암호동기화

SFU 는 UNIX 를 위해 Windows 통과암호를 동기화한다. 우의 실례에는 통과암호정보가 포함되여 있는 Windows 로부터 UNIX 에로 보내는 파일이 있다. 파일은 root 를 위해 읽기방식으로만 설정하게 되여 있다. 이것은 통과암호를 동기화하는 도구로서의 기법이다. 설치를 끝낸 후 사용자는 Windows 상의 통과암호와 UNIX 상의 통과암호를 동기화할것이다.

제32장. Samba

Samba에 대한 개괄

Samba는 UNIX 호스트가 Windows 체계에 대한 파일봉사기와 같이 동작하게 하는 응용프로그람이다. Windows 체계들은 그것의 본래 Windows 망기능으로 UNIX 파일체계와 인쇄기들을 처리할수 있다.

이 장에서는 Samba의 설치에 대한 개괄과 그 기능의 일부를 보여 준다. 그중에서도 파일봉사기기능에 중심을 둘것이다. 또한 원격 UNIX 체계상에서는 Windows 체계에서 구동기문자나 그림기호들과 같이 나타나는것들에 대하여 공유하게 할것이다. 왜냐하면 Linux 체계에서 설치하고 실행하는 이 책의 전반에서 사용된 RED Hat Linux 프로그람과 함께 Samba가 제시되기때문이다. Samba는 대다수 UNIX 판본들에서 리용한다.

Samba는 Server Message Block(SMB)규약을 사용하여 이 파일공유기능을 제시한다. SMB는 TCP/IP상에서 실행한다. 이 장의 실례에서 Windows체계와 UNIX체계는 TCP/IP와 SMB를 실행하고 있다. 이것들은 두 체계사이에 공유파일을 설정하기 위하여 요구되는 모든 기술을 제시하여 준다.

16장에서는 UNIX 체계상에서 파일들을 Windows 체계에서 실행하는 Network File System(NFS)을 서술하였다. 이 기능은 이 장에서 서술하는 Samba와 류사하다. 파일공유기능을 보충하는데서 Samba도 공유인쇄기를 주고 보충된 사용자는 조종처리를 한다. 이 장에서는 기능들에 대하여 중심을 두지 않기로 하겠다. 그렇지만 Samba는 이 분야에서 일정하게 발전된 기능을 제시하고 있다.

이 기능을 서술하는 동안 Samba는 파일공유, 인쇄기공유기능을 제시하며 발전된 사용자가 파일조종을 처리하는 기능을 제시한다. 보통의 프로그람처럼 Samba와 GNU Public License(GPL)밑에서 보급되는 프로그람이 많이 개발되였다. 왜냐하면 프로그람이 유연하며 많은 업체들이 프로그람을 발전시키기 위하여 노력하고 많은 시간을 바치고 있기때문이다. 그러므로 사용자는 Samba와 다른 프로그람의 추가적기능을 얻어 내려고 하고 있다. 이 장의 마지막부분에 도달하면 Samba와 다른 유연한 프로그람에 대하여 더 많은 정보를 얻을것이다.

설치

Samba가 Red Hat Linux CD-ROM으로 보급되기때문에 단순한 Samba의 **설치**(Set Up)에서는 Red Hat Linux를 사용하여야 한다. Red Hat Linux를 설치할 때 사용자가 모든

UNIX 판본을 취할수 있기때문에 적재할것을 요구하는 쏘프트웨어를 선택할수 있다. 사용자가 처음 조작체계를 적재할 때 Samba 의 적재를 할수 없으면 Samba 나 다른 프로그람을 적재하는 지령행에 Gnome RPM 또는 rpm 을 사용할수 있다. 이 도구들은 체계조직을 취급한 장에서 간단히 론의되였다. Linuxconf 를 사용하여 Samba 를 설치하기 위해서는 일부 단순한 과제들을 실행하여야 한다.

그림 32-1 에서는 3 개의 디스크로 Samba file Server 밑에서 linuxconf window Disk Shares 의 공유에 대하여 보여 준다.

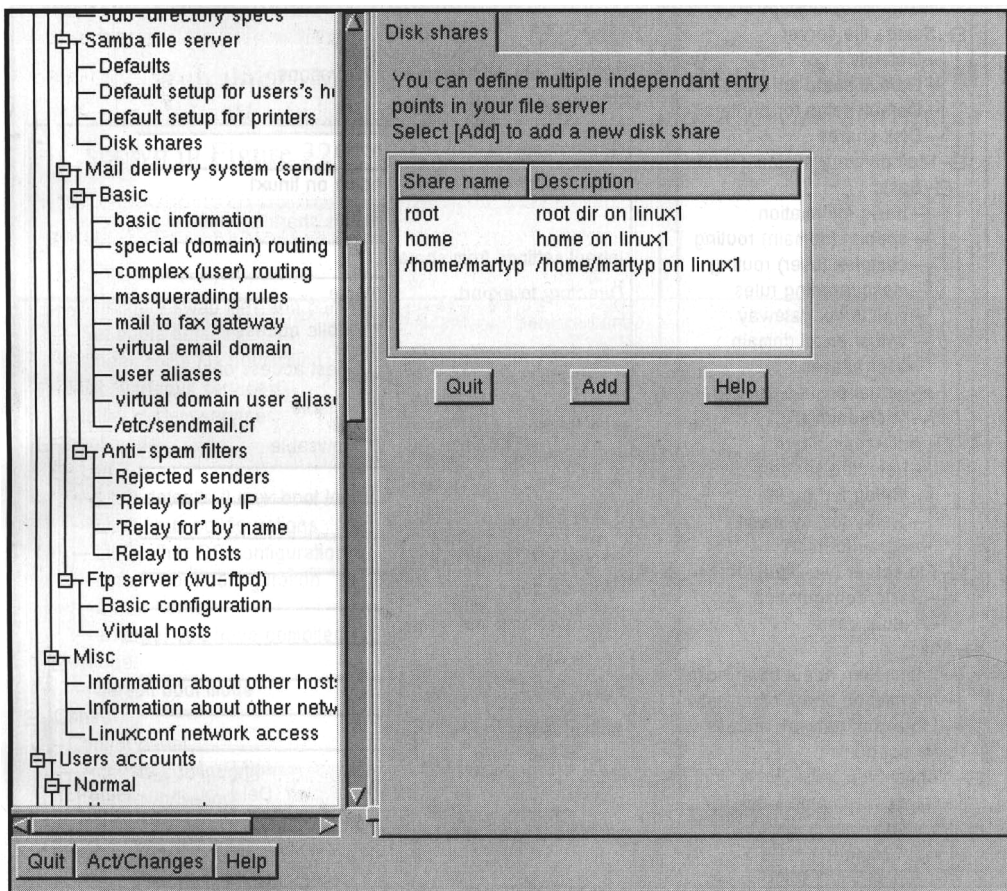



그림 32-1. Linuxconf 에서 공유정보

SMB 를 실행하는 다른 체계들을 유용하게 만든 Linux 체계에는 3 개의 공유이름을 가지고 있다. 그림 32-2 는 home 이라고 불리우는 중간공유과정에 대한 더 구체적인 통보를 보여 주고 있다.

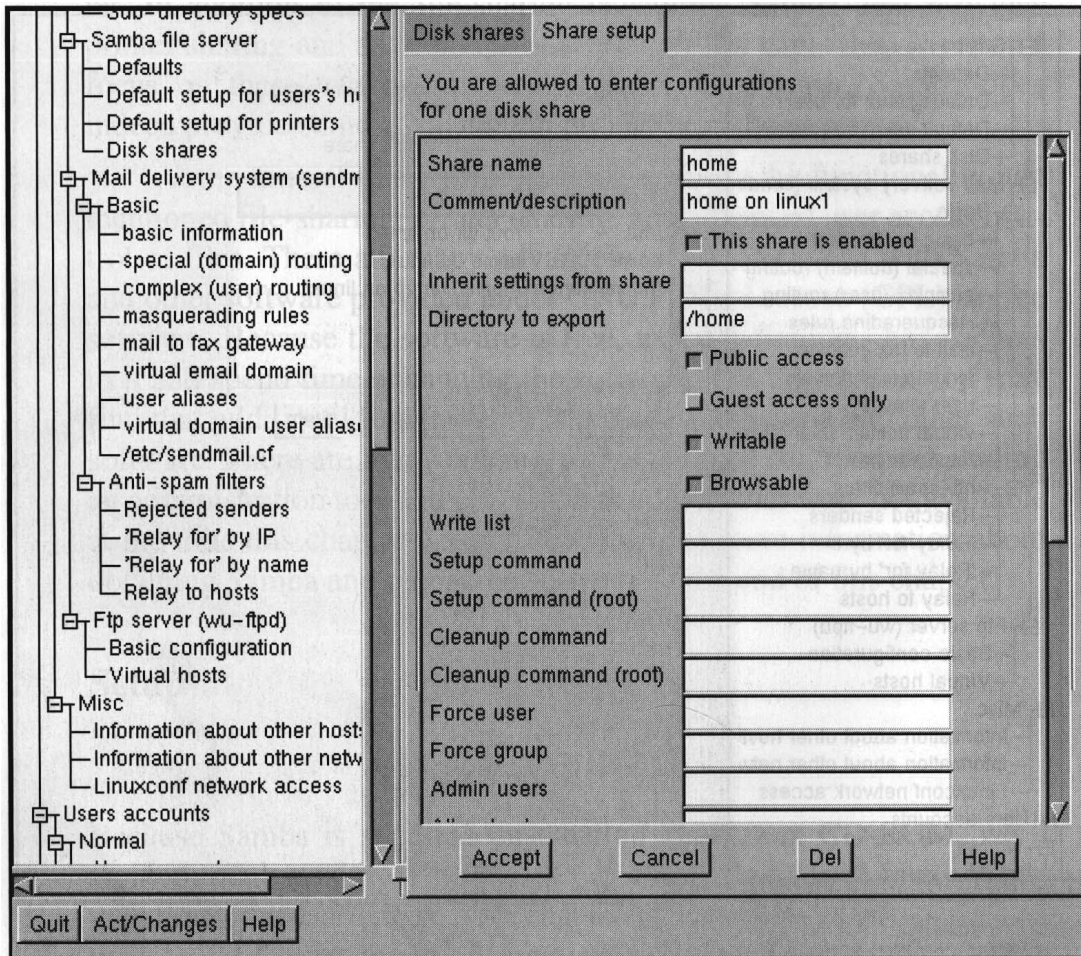



그림 32-2. Linuxconf 에서 /home 공유정보

이 공유는 Linux1 에서의 /home 등록부이다. 그림에서 4 개 선택항목들은 허락과 련관되여 있다. 이 공유에서 Public access, Writable, Browsable 을 허용하였다. 그것은 Guestonly access 에서 제한하지 않았다. 이 공유에서 실지로 사용자는 공유하기 위하여 알맞는것을 할당할것을 요구한다.

3 개의 공유를 할당하고 smb 로 시작하기 위해서는 Linuxconf 를 리용해야 한다. 그림 32-3 에서 제시된것처럼 Control service activity 에서 smb 가 가능하므로 그렇게 한것이다.

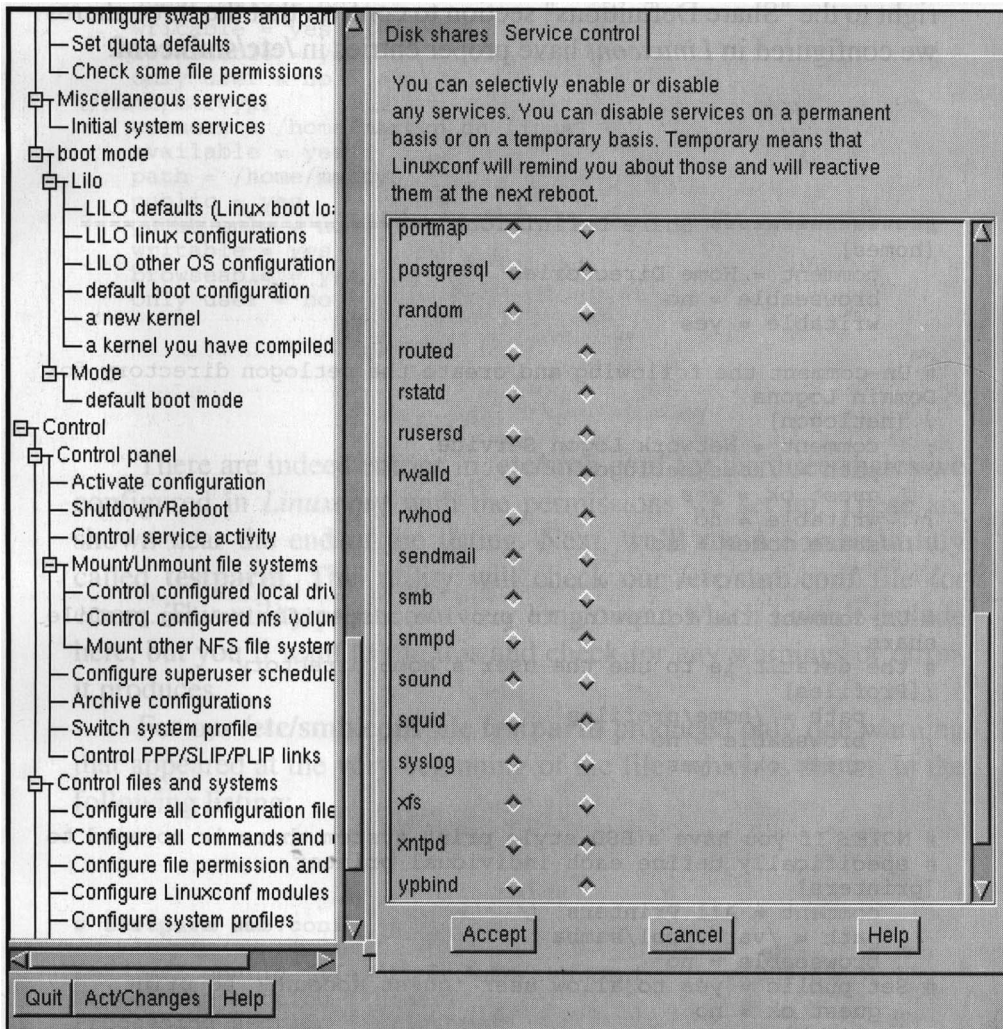



그림 32-3. Linuxconf 의 Control service activity 에서 smb 시동

smb 는 체계가 시동될 때 봉사시작을 하게 한다.

smb 를 사용하기전에 Linuxconf 로 실행한 작업을 빨리 검열하는 련결을 할수 있다. 사용자가 한 모든것을 Linuxconf 로 수동적으로 완성할수 있다.

첫째 검사는 파일 /etc/smb.conf 에 대한 검사이다. 이것은 smb 의 모든 구성통보를 포함한 파일이다. 이제 Linuxconf 에서 구성한 3 개 공유들을 /etc/smb.conf 에 부합되게 구성하기 위하여 "Share Definition"부분을 보자.

```
# ================== Share Definitions ==================
[homes]
   comment = Home Directories
   browseable = no
   writable = yes
```

```
# Un-comment the following and create the netlogon directory for Domain Lagons
;   [netlogon]
;   comment = Network Logon Service
;   path = /home/netlogon
;   guest ok = yes
;   writable = no
;   share modes = no

# Un-comment the following to provide a specific roving profile share
# the default is to use the user's home directory
;[Profiles]
;   path = /home/profiles
;   browseable = no
;   guest ok = yes

# NOTE: If you have a BSD-style print system there is no need to
# specifically define each individual printer
[printers]
          comment = All Printers
          path = /var/spool/samba
          browseable = no
# Set public = yes to allow user guest account' to print
guest ok = no
writable = no
printable = yes
   [root]
          comment = root dir on linux1
          available = yes
          path = /
          public = yes
          guest only = no
          writable = yes
          browseable = yes
          only user = no
   [home]
          comment = home on linux1
          available = yes
          path = /home
          public = yes
```

```
        guest only = no
        writable = yes
        browseable = yes
        only user = no
[/home/martyp]
        comment = /home/martyp on linux1
        available = yes
        path = /home/martyp
        public = yes
        guest only = no
        writable = yes
        browseable = yes
        only user = no
```

우리가 설치한 허락과 함께 Linuxconf 에서 구성한 3 가지의 공유에 대한 기입은 /etc/smb.conf 에 있다. 이것들을 목록의 끝부분에서 보여 주고 있다. 다음 testparm 이라는 Samba 봉사프로그람을 실행한다. 이 봉사프로그람은 /etc/smb.conf 파일이 오유가 있는가를 검사한다. 이 봉사프로그람은 이 부분에서 설명하지 않는 대단히 긴 출구정보를 내보낸다. 그러나 사용자는 제시된 임의의 경고나 오유에 대한 검사와 실행을 요구할수 있다.

/etc/smb.conf 의 testparm 파일은 파일이 시작될 때 나타나는 한개 경고만 제시하기로 한다. 이것은 다음 목록에서 제시되였다.

```
# testparm smb.conf

Load smb config files from /etc/smb.conf
Processing section "[homes]"
Processing section "[printers]"
Processing section "[root]"
Processing section "[home]"
Processing section "[/home/martyp]"
Loaded services file OK.
WARNING: You have some share names that are longer than 8 chars
These may give errors while browsing or may not be accessible to some older clients
Press enter to see a dump of your service definitions
                                    ⟨
```

testparm 은 공유이름이 긴데 대하여 경고를 제시하고 있는데 이 testparm 은 순서대로 되여 있는 3 개의 공유이름목록을 포함한다. testparm 은 긴 공유이름경고를 등록하지 않는다. 왜냐하면 실례에서 사용된 Windows 체계가 긴 이름들을 처리할수 있기때문이다.

그렇지만 DOS 체계에서라면 이 이름은 문제로 되였을것이다. 간단하게 Linux 에 의하여 량립할수 없는 가능한 이름들을 어떻게 주소화하는가에 대하여 보기로 하자.

다음목록에서 보는바와 같이 다음에 하려고 하는 검사는 smbd 라고 부르는 Samba daemon 이 실행한다.

```
# ps -efl | grep smbd

140  S  root   490     1  0  60  0   -  553   do_sel  Sep28   ?      00:00:00  [smbd]
140  S  root  1493   490  0  60  0   -  896   do_sel  Sep29   ?      00:00:00  smbd -D
000  S  root  1976  1951  0  70  0   -  288   pipe_r  09:09   pts/1  00:00:00  grep smbd
```

이 ps 출구는 Smbd 가 실행되고 있다는것을 보여 준다. 다음은 아래실례에서 보여주는것처럼 "LISTEN"에서 netbios-ssn 봉사가 실행되는것을 확인하는 검사를 하자.

```
# netstat -a | grep netbios

tcp   0   0   linuxi:netbios-ssn    f4457mfp2:1047   ESTABLISHED
tcp   0   0   *:netbios-ssn         *:*              LISTEN
udp   0   0   linux | netbios-dgm   *:*
udp   0   0   linuxl:netbios-ns     *:*
udp   0   0   *:netbios-dgm         *:*
udp   0   0   *:netbios-ns          *:*
```

이 목록이 출구된것은 netbios 가 실행되고 있다는것을 보여 준다. Samba 봉사기에 련결할 때까지 볼수 없는 ESTABLISHED 련결이 표시된다. 이 장에서는 실례들을 준비하여 접속하게 하였다.

Linux 체계에서의 Windows 체계상에서 파일접근을 위한 Samba 의뢰기를 사용하자. 지금까지는 이 기능을 취급하지 않았으나 Linux 체계가 다른 기능이 없고 파일봉사기처럼 동작하기때문에 많은 유용한 정보를 주는 봉사프로그람 smbclient 를 리용한다. 다음의 실례를 보면서 smbclient 봉사프로그람으로 Samba 의 설치에 대한 전반적인 리해를 하기로 하자.

```
# smbclient -L linuxl

Added interface ip=192.168.1.1    bcast=192.168.1.255    nmask=255.255.255.0

Domain=[MYGROUP]    OS=[Unix]    Server=[Samba 2. 0. 3]

        Sharename      Type      Comment
        -------------  -------   -------------
        root           Disk      root dir on linux1
        home           Disk      home on linux1
        /home/martyp   Disk      /home/martyp on linux1
        IPC$           IPC       IPC Service (Samba Server)
        lp             Printer
```

Server	Comment
---------	---------
LINUX1	Samba Server

Workgroup	Master
---------	---------
ATLANTA2	F4457MFP2
MYGROUP	LINUX1

이 봉사프로그람은 우리가 설치하는 3 개의 공유에 대하여 포함하는데 실례에서는 Samba 봉사기 및 다른 유용한 통보들을 포함하여 Samba 설치의 유용한 개요를 제시한다.

봉사기상에서는 Samba 시험을 계속할수 있다. 그러나 Samba 가 실행될 때마다 매번 목록을 제시하여 준다. Windows 의뢰기에 가서 Samba 봉사기상에서 유용한 공유에 대하여 처리하기 위하여 explorer 를 사용하자.

공유사용

explorer 를 사용하여 망구동기를 만든다. 그림 32-4 에서 제시된것처럼 Samba 봉사기상에서 /root 를 Windows 체계상의 E:구동기에 만들자.

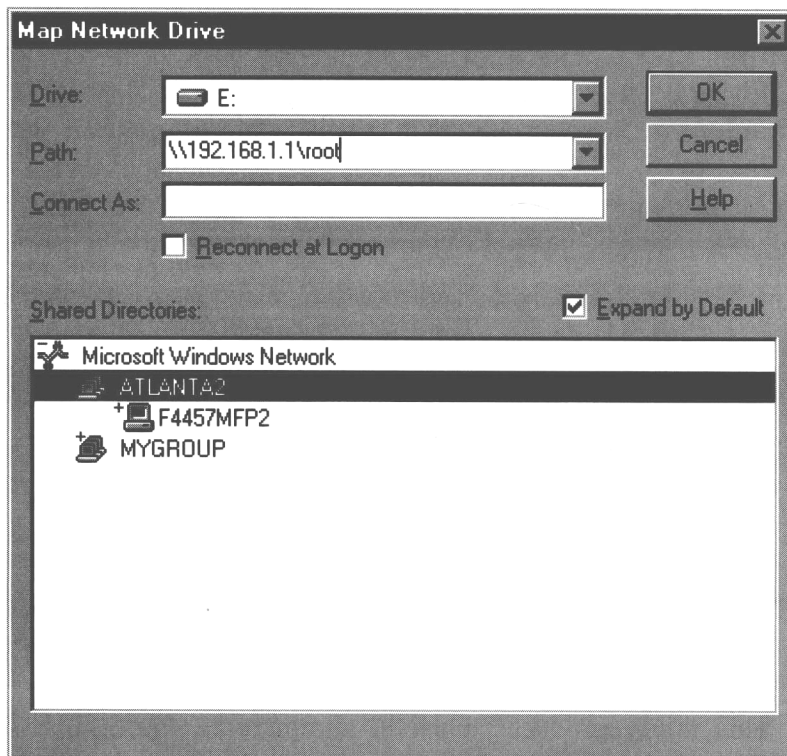



그림 32-4. 구동기 E:에서 /root 만들기

PC 상의 지령행에서 net use E:\\192.168.1.1\root 를 리용하여 망구동기를 만들수 있다. explorer 를 리용하여 Linux1 상의 /로 만들어 진 E 구동기를 완전히 처리한후에 Windows 체계상의 이 구동기를 처리할수 있다.

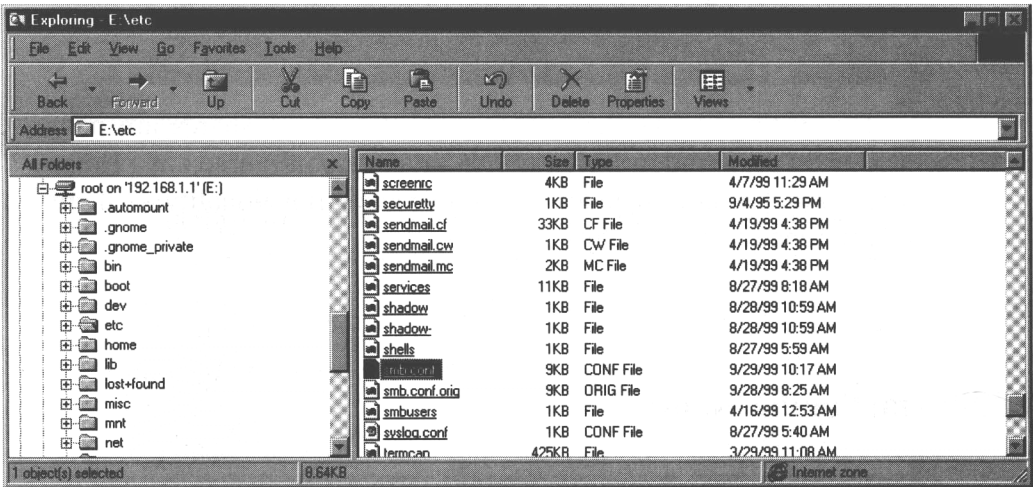



그림 32−5. Windows 체계에서 E:\etc 보기

그림 32−5 에서 Linux1 에서의 /etc 등록부를 교체하고 smb. conf 파일을 선택하자. 이 실례를 통해서 Linux1 에서 /이 Windows 체계상에서 다음실례가 보여 주는것처럼 Windows 체계상으로 완전히 접근한다는것을 알수 있다.

c: **net view \\192. 168. 1. 1**

Shared resources at \\192. 168. 1. 1

Samba Server

Share name	Type	Used as	Comment
/home/martyp	Disk	G:	/home/martyp on linux1
home	Disk	F:	home on linux1
lp	Print		
root	Disk	E:	root dir on linux1

The command completed successfully.

이 실례는 Linux1 의 모든 3 개 공유가 Windows 체계상에서 리용할수 있다는것을 보여 준다. 여기서는 어떤 인쇄기공유도 구성하지 않았다. 이것도 역시 Samba 기능의 부분과 같이 포함된것이므로 어떤 인쇄기련결도 하지 않았다.

보충적인 Samba 기능

Samba Web 구성도구(SWAT)

SWAT 는 Samba 를 위한 Web 에 기초한 조직도구이다. 대부분 Samba 구성과제에 대한 대면부를 구성하고 준비하는것은 어렵지 않다. Red Hat Linux 체계상에서 다음의 걸음들은 SWAT 를 실행해야 한다. 사용자가 서로 다른 UNIX 판본을 가지고 있다면 그때 걸음은 빠를것이다. 다음행은 /etc/services 을 확인한다.

```
swat 901/tcp
```

/etc/inetd.conf 에 있는 다음행은 설명하지 않기로 한다. 이 행은 이미 적재된 Linux 조작체계의 부분처럼 파일속에 있다.

```
swat stream tcp nowait. 400 root/usr/sbin/swat swat
```

지령의 앞부분은 Process ID(PID)에 의한 inetd 처리를 없애고 그다음 그 PID 지령을 없애는것을 보여 준다.

ps-ef | grep inetd
kill ?1 PID of inetd

이 경우에는 열람기대면부에서 SWAT 를 실행할수 있다. Samba 봉사기의 IP 주소와 901 인 포구번호를 다음과 같이 지적한다.

netscape http://192. 168. 1. 1:901

열람기는 SWAT 에서 사용자가 SWAT 구성을 위한 수정을 할수 있다는것을 확인하기 위해 SWAT 가 동작할 때 사용자이름과 암호를 요구한다. 그림 32-6 은 SWAT 대면부를 보여 준다. 그림은 HOME, GLOBALS, SHARES, PRINTERS, STATUS, VIEW, PASSWORD 에 대한 련결을 포함한다.

그림 32-6 에서 SHARES 를 선택하여 SWAT 에서 먼저 구성한 3 개 항목을 리용할수 있다. SWAT 는 Samba 를 구성하는 좋은 대면부이다. 그러나 이 장의 앞에서 처리한 수동처리에 대한것은 알아 낼수 없다.

SWAT 로 생겨 난 결과를 교체하기 위해서는 다음 지령으로 smbd 를 재기동해야 한다.

/etc/rc.d/init.d/smb restart

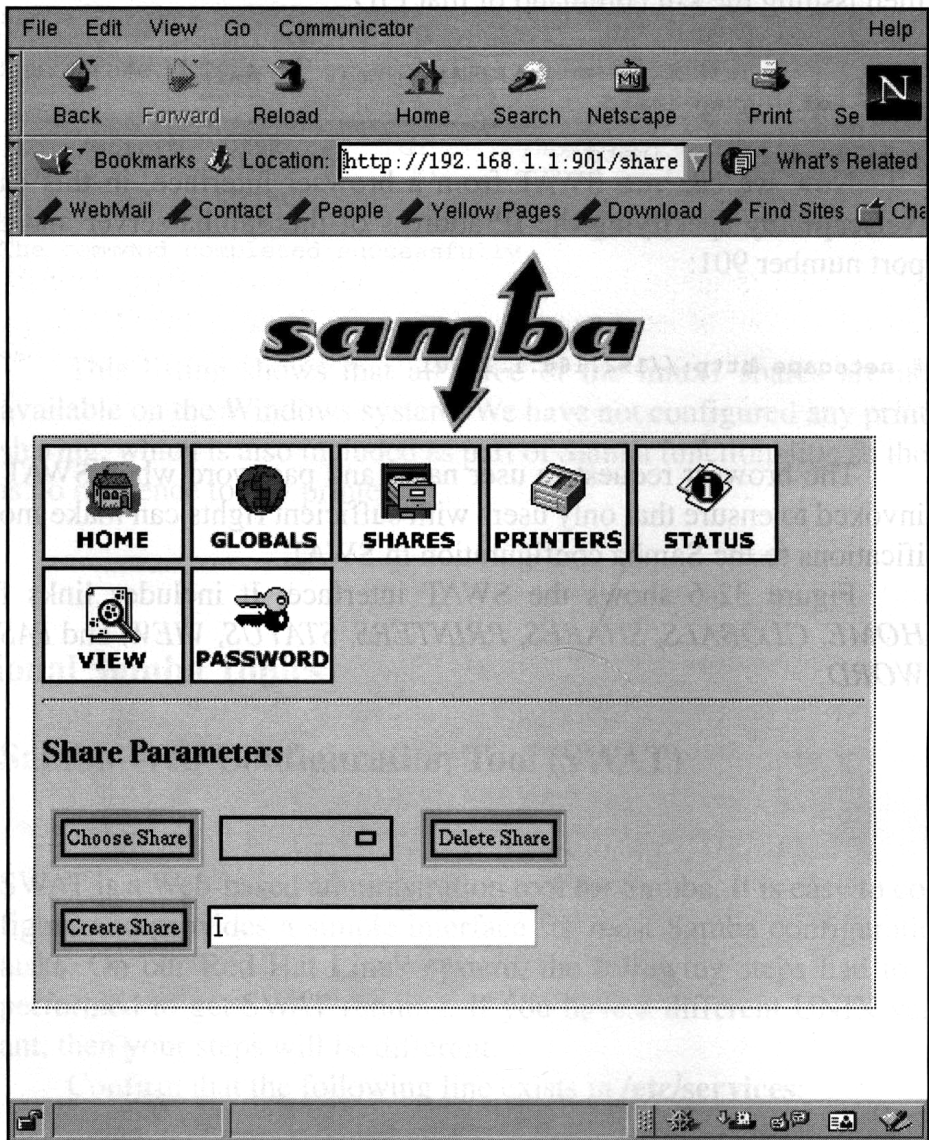



그림 32-6. SHARES 가 선택된 SWAT

Linux 체계상에는 SWAT 와 그것의 기능을 서술한 많은 자료가 있다. 그림 32-7 은 이 장에서 취급하면서 많이 론의한 자료와 함께 SWAT 에서 HOME 페지를 보여준다.

모든 Samba 와 관련된 프로그람에 대한 많은 직결자료가 www.samba.org 에 제시되여 있다.

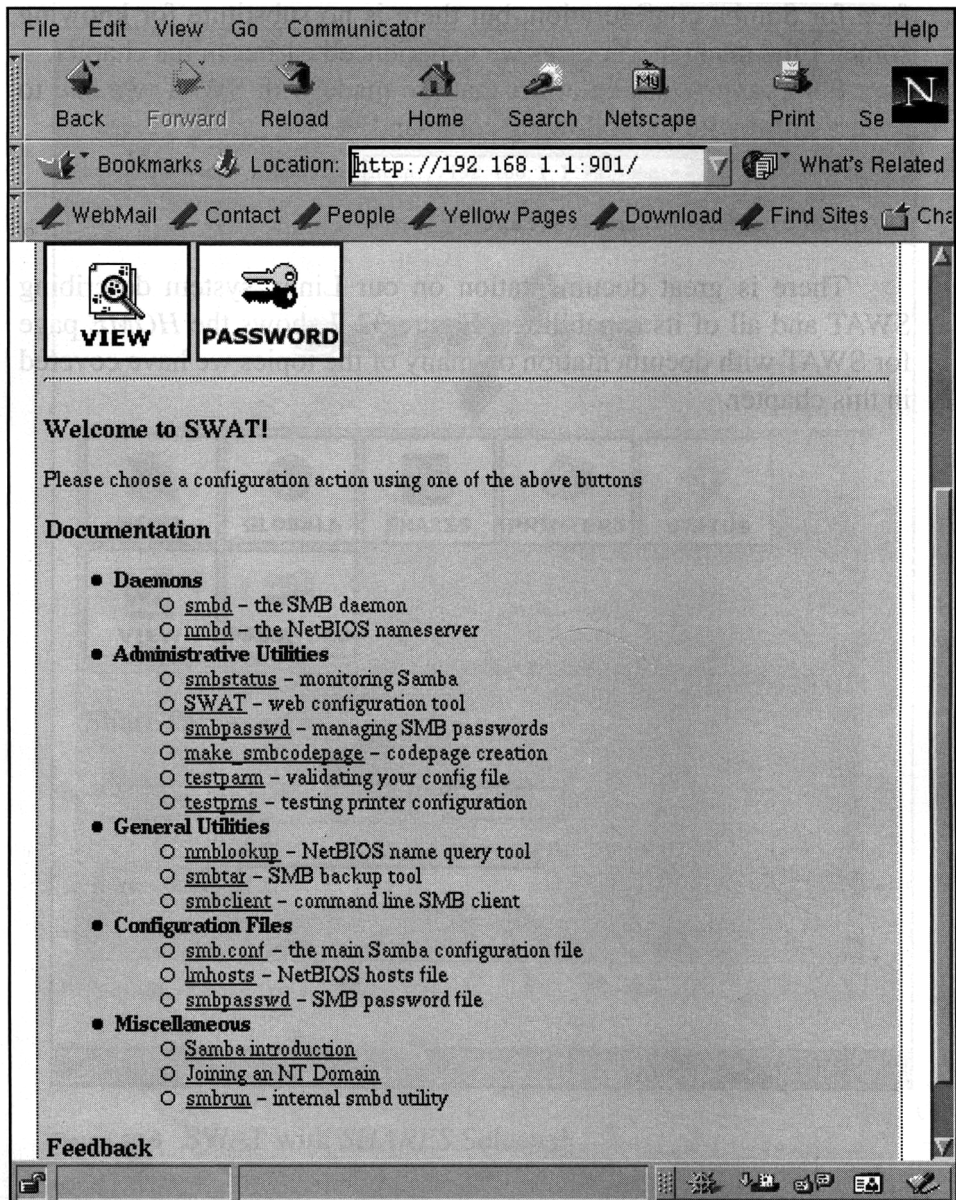



그림 32-7. 쓸모 있는 자료를 보여 주는 SWAT HOME

일지파일

대부분의 UNIX 응용프로그람과 류사한 Samba는 많은 가입을 하게 한다. smb.conf 파일은 사용자가 Samba의 가입을 위하여 필요한 위치를 지적하는 부분을 포함한다. 아래의 간단한 부분은 사용자가 매 Windows 련결장치를 위하여 개별적인 일지파일을 가질수 있다는것을 보여 주며 이때 사용자는 일지파일의 최대크기를 지적할수 있다.

```
# this tells Samba to use a separate log file for each machine
# that connects
   log file=/var/log/samba/log．%m

# put a capping on the size of the log files (in Kb).
   max log size=50
```

f4457mfp2 가 이 장의 실례에서 사용된 Windows 체계일지파일을 포함하므로 /var/log/samba 등록부는 Samba 일지파일들을 포함한다.

```
# ls -1 /var/log/samba

total 14
-rw-r--r--  1  root  root  1915  Sep  30  09:29  log．f4457mfp2
-rw-r--r--  1  root  root  213   Sep  29  10:14  log．linux1
-rw-r--r--  1  root  root  0     Sep  29  04:02  log．nmb
-rw-r--r--  1  root  root  8234  Sep  28  12:25  log．nmb．1
-rw-r--r--  1  root  root  3215  Sep  28  12:19  log．smb
-rw-r--r--  1  root  root  0     Oct  1   11:17  smbconf3．txt
```

파일이름자르기

많은 경우에 Windows 와 UNIX 사이에는 존재하는 파일이름들이 량립될수 없다. 사용하는 Windows 판본에 의존하면 UNIX 와 량립할수 없는 많은 파일이름이 있다.

그림 32−8 은 선택된 파일 gnome_private 와 explorer 창문인데 이것은 Linux 봉사기상우에 있다. WindowsNT 체계상에서 이 파일이름은 fine 으로 나타난다. 그렇지만 이 그림에 있는것은 DOS 체계에 있는 경우에 이 파일을 원만하게 변화시킬수 있는 DOS 이름을 보여 주는 속성창문이다.

Windows 체계인 경우에 그것이 Linux 체계에 나타날 때 .gnome_private 파일을 조종하면 문제가 제기되지 않는다. DOS 체계인 경우에는 "mangling"이라는 이름을 가진 파일이 생길것이다. 8.3 형이름만을 사용하는 DOS 는 8 개 문자의 이름과 3 개 문자의 확장자를 가진다.

이 실례에 있는것과 같이 점으로 시작한 Samba 간략파일들은 초기파일이름을 하쉬알고리듬을 사용하여 leading 점파일들을 제거하고 처음 5 개 문자를 쓰고 다음에 기호 ～을 표시하며 마지막 2 개 문자를 붙인다. 파일이름은 간략하여 모두 8 개의 기호를 가진다. 모든 문자들은 대문자이다.

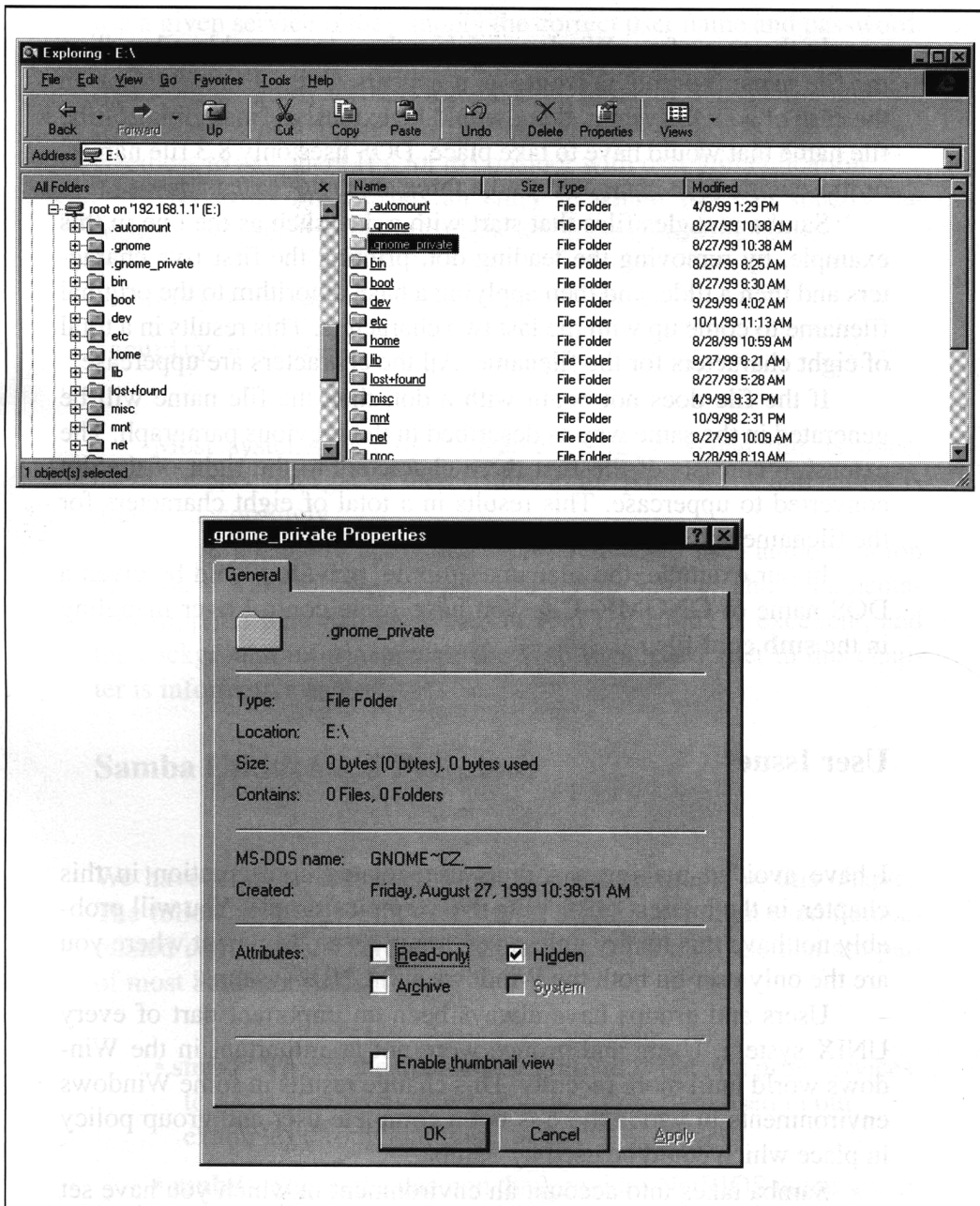



그림 32-8. Explorer 와 Mangled 이름에 있는 .gnome_private

앞단락에서 서술한것처럼 파일이 점으로 시작하지 않으면 파일이름을 같은 방법으로 만든다. 확장자는 대문자로 전환한 점의 오른쪽 첫 3 개 문자를 포함한다. 이렇게 하여 파일이름은 8 개의 파일이름과 3 개의 확장자로 구성된다.

실례에서 파일이름 .gnome_private 는 DOS 에서 GNOME~CZ 로 나타날것이다. 사용자는 smb.conf 파일에서 조종한다.

사용자문제

이 장에서는 우와 같은 실례를 계속 고찰하지만 사용자와 깊은 련관관계를 구성하고 있는것은 고찰하지 않기로 한다. 사용자는 Windows 와 UNIX 두 체계에서 작업해야 만족할것이다.

사용자와 그룹들은 언제나 매 UNIX 체계의 중요한 부분이다. 사용자와 그룹들이 중요하게는 최근에까지 Windows 세계에 있지 않았다. 이것은 Samba 를 사용하는 위치에 사용자와 그룹의 방법이 전례 없는 일부 Windows 환경을 변화시킨다.

Samba 는 Windows 사용자와 그룹을 설치하고 Windows 사용자와 그룹의 관계를 보여주는 환경계산서를 만든다. Samba 에서 사용자표식의 인정은 그들이 정확한 사용자이름과 암호를 입구하는 경우에 의뢰기가 봉사기를 사용할수 있게 설치한다. 공유표식의 인증은 UNIX 체계우에서 "guest account"의 처리를 허용하여 만든다. 이것은 이전에 의뢰기가 사용자이름과 암호를 사용하지 않았을 때에 해당된다. 말하자면 사용자인증과 함께 많이 고려된다.

다음실례에서 보여 주는것처럼 smb.conf 파일은 "user"혹은 "share"와 같이 필요한것을 렬거할수 있다.

security=user
security=share

대부분의 체계들은 사용자표식보안을 쓰고 있다. 이것이 처리될 때 사용자는 passwd 파일과 허용된 처리에서 이름을 검열 받는다.

Samba 를 설치하면 사용자가 검열할것이 필요한 사용자인증과 관련한것들을 보충적으로 알수 있다. Red Hat Linux 부분과 같은 Samba 와 관련된 필요한 자료는 많이 제시되여 있다. Web 싸이트의 배경정보는 이 장의 마지막에 제시되여 있다.

Samba 봉사프로그람과 프로그람들

이 장에서는 여러가지 Samba 봉사프로그람과 프로그람을 사용한다. 아래에서는 Samba 와 관련하여 흔히 리용하는 지령들을 서술한다. 다음의 내용은 대부분의 Samba 설치부분에 대한 안내페지이다.

- smbd — 이것은 이 장에서 Windows 체계를 실례로 사용한것처럼 SMB 의뢰기에 대한 파일과 인쇄봉사를 주는 기법이다.
- nmbd — 이것은 NetBIOS 이름봉사기능력을 주며 열람하게 하는 기법이다.
- smbclient — 이것은 봉사기가 다른 봉사기에서 원격으로 SMB 를 설치하기 위한 처리를 하는 프로그람이다.
- testparm — 이것은 /etc/smb.conf 에 대한 검사프로그람이다.

- smbstatus 이것은 현재 Samba 련결에 대한 통보를 표시하는 프로그람이다.
- smbpasswd 이것은 국부장치에서 사용자 SMB 암호를 변화시키기 위하여 사용하는 프로그람이다.
- smbrun 이것은 smbd 에 대한 쉘지령을 실행하는 프로그람이다.
- smbtar 이것은 UNIX 테프구동기에 대하여 SMB 공유들이 제자리로 들어 오게 하는 프로그람이다.
- smbmount 이것은 SMB 파일체계를 태우는 프로그람이다.
- smbumount 이것은 SMB 파일체계를 태운것을 해제하는 프로그람이다.

Samba 와 관련한 지령을 주는 직결설명서에서는 더 구체적인 설명을 주고 있다. 이 장에서 실행된 단순한 설치까지도 사용자가 이 프로그람의 일부를 실행하여야 한다.

Samba 의 입수방법

이 장에서 사용된 실례에서는 Red Hat Linux CD-ROM 의 부분처럼 설치한 Samba 를 Linux 체계우에서 설치한다. 사용자의 UNIX 판본에 Samba 가 없을 때 Samba 를 적재하려한다면 Samba 를 Web 에서 얻는다.

www.samba.org 는 시작하기 위한 장소이다. Web 싸이트에서 사용자는 자기 나라에서 "download site"를 선택할수 있다. 사용자는 Samba 의 정보가치를 주는 www.samba.org 에서 "Web sits"를 선택할수 있다.

이 장에서 먼저 언급한 GNU General Public License 들을 포함하여 Samba 와 관련한 Web 싸이트에는 가치 있는 정보가 있다.

이 장에서 먼저 보여 주고 사용한 프로그람들의 상세한 설명들을 포함하여 Samba 를 밑으로부터 적재하려고 결심하였다면 사용자는 아마 체계상에서 이미 번역된 Samba 를 적재하거나 Samba 를 만들거나 콤파일하는 선택항목을 준다. 사용자는 많은것을 고려하여 선택한다. 이 장에서 Red Hat Linux 로 작업할 때처럼 Samba 배치를 할수 있으면 이미 콤파일된 Samba 를 리용하다. Samba 가 어떻게 작업하고 구성되였는가를 더 배우려고 하고 최신판과 제일 높은 판본이 요구되면 사용자가 원천을 적재하고 그것을 콤파일하여야 한다.

이 장에서 고찰한것처럼 사용자가 큰 Samba 체계를 가지고 있으면 유용한 많은 자료를 보기 위하여 Samba 와 련관된 Web 싸이트를 보는것이 좋다.

색 인

ㄱ

ㄴ

ㅅ

ㅈ

ㅇ

ㅃ

ㅆ

* * *